НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

____ _____ Теплоенергетичний факультет _____

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____2019р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему: Система інтеграції елементів віртуальної реальності в освітню Web-систему

Виконав: студент 4 курсу, групи ТІ-51

_____Мінаєв Кирило Владиславович_____ _____
(прізвище, ім’я, по батькові) (підпис)

Керівник Шалденко Олексій Вікторович_____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту
Мінаєва Кирила Владиславовича
(прізвище, ім’я, по батькові)

1.Тема роботи Система інтеграції елементів віртуальної реальності в освітню Web-систему
Керівник роботи Шалденко Олексій Вікторович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ” ” 201 р. № _____

2. Строк подання студентом роботи _____
3. Вихідні дані до роботи: мова програмування C#.
4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Розробити систему інтеграції елементів віртуальної реальності в освітню Web-систему. Розробити програмний інтерфейс платформи.

5. Перелік ілюстративного матеріалу: схеми архітектури додатку, діаграма прецедентів системи, діаграма структури системи, зразки розробленого інтерфейсу додатку.
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання ” ”_____201 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі		
2	Розробка архітектури та загальної структури системи		
3.	Розробка структур окремих підсистем		
4.	Програмна реалізація системи		
5.	Оформлення пояснювальної записки		
6.	1 Захист програмного продукту		
7.	2 Передзахист		
8.	Захист		

Студент _____ Мінаєв К.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шалденко О.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

бакалаврської дипломної роботи Мінаєва Кирила Владиславовича на тему «Система інтеграції елементів віртуальної реальності в освітню WEB-систему»

У роботі розглядаються питання розробки системи інтеграції елементів віртуальної реальності в освітню систему.

Мета роботи – реалізувати таку систему інтеграції елементів віртуальної реальності в освітню систему, щоб нею було зручно користуватися, була можливість легко додавати нового функціоналу та щоб ця система була невід'ємною частиною освітньої WEB-системи. Після реалізації системи інтеграції елементів віртуальної реальності протестувати цю систему на наявність помилок.

У процесі роботи досліджувалися програмні продукти та технології, які застосовуються для створення систем інтеграції елементів віртуальної реальності.

Результатом роботи є працююча та протестована система інтеграції елементів віртуальної реальності в освітню WEB-систему, опис процесу побудови системи.

Розробка системи інтеграції елементів віртуальної реальності була здійснена за допомогою ігрового двигуна Unity3D та у середовищі IntelliJ Rider.

Ключові слова: ЕЛЕМЕНТИ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ, ІНТЕГРАЦІЯ, UNITY3D, VIRTUAL REALITY

ABSTRACT

Bachelor diploma work by Minaiev Kyrylo "Virtual reality elements integration system into the educational WEB-system"

The paper deals with the development of a system for integrating elements of virtual reality into the educational system.

The purpose of the work is to implement such a system of integration of elements of virtual reality into the educational system so that it was convenient to use, it was easy to add a new functional and that this system was an integral part of the educational WEB-system. After realizing the system of integration of elements of virtual reality, test this system for errors.

In the course of work, software products and technologies that were used to create systems for integrating virtual reality elements were studied.

The result of the work is a working and tested system of integration of elements of virtual reality into the educational WEB-system, a description of the process of building a system.

The development of a system for integrating virtual reality elements was implemented using the Unity3D gaming engine and the IntelliJ Rider environment.

Key Words: ELEMENTS OF VIRTUAL REALITY, INTEGRATION, UNITY3D

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

VR	–	віртуальна реальність
HDM	–	монітор високої розподіленої здатності
SDK	–	бібліотека
LDRP	–	рендеринг зображення
Шейдер	–	програма для відеокарти
Спрайт	–	картинка
Плагін	–	розширення функціональності
Рендеринг	–	створення

ВСТУП

Із кожним днем все більше сучасних технологій інтегруються в різні сфери нашого життя. Однією із цих сфер є сфера освіти, яка кожен день розвивається, вчені, вчителі та викладачі придумують нові методи для покращення рівня освіти. Тому залучення сучасних технологій лише сприяє тому, що люди починають вчитися інакше, із більшим поглинанням знань, використовуючи методи, недоступні раніше.

Велика кількість різноманітних методів вже створено із використанням комп'ютерів, планшетів, смартфонів. Та можливості збільшуються із кожним днем, адже постійно на світ з'являються нові освітні застосунки. Університети, школи роблять величезні замовлення для створення різноманітних додатків для поліпшення рівня викладання освіти.

Одним із найважніших пунктів у навчанні є практика. Адже без неї теорія не має сенсу. Але іноді практику не можна проходити одразу після набуття теоретичних знань, бо вона небезпечна для людини без досвіду, або дуже дорога.

Актуальність цієї проблеми полягає в тому, що в людства є запит на нові знання, з'являються нові напрямки освіти, з'являються нові методи візуалізації інформації, наші знання стають масштабнішими і глибшими, а традиційні навчальні методи вже не можуть забезпечити людину необхідними знаннями. Особливо етап практики є дуже важливим, але вона може бути небезпечною для життя, може коштувати багато коштів, може бути дуже складною для відтворення, або і зовсім недосяжною. В багатьох сферах людської діяльності людину, яка тільки пройшла курс навчання, не можна одразу допустити до повноцінної практики.

Метою дипломної роботи є реалізація такої системи інтеграції елементів віртуальної реальності в освітню систему, яка сприятиме підвищенню рівня навчання окремої сфери, допоможе проводити практичні зайняття без ризику для здоров'я та особливих витрат. Після реалізації системи інтеграції елементів віртуальної реальності протестувати цю систему на наявність помилок та надати рекомендації по роботі з системою інтеграції елементів віртуальної реальності в освітню WEB-

систему.

Головними задачами дослідження є:

1. Вивчення й аналіз існуючих систем, що дозволяють інтегрувати елементи віртуальної реальності в навчальний процес.

2. Вивчення й аналіз новітніх підходів інтегрування елементів віртуальної реальності в освітній процес.

3. Порівняльний аналіз конкурентних систем та практик.

4. Розробка власного варіанту системи інтегрування елементів віртуальної реальності на базі найкращих практик, підходів та інструментів.

Впровадження запропонованої системи інтеграції елементів віртуальної реальності відкриває можливість впровадження віртуальної реальності у освітній процес та використовувати її як один із способів підвищення якості навчального процесу.

1. ПОСТАНОВКА ЗАДАЧІ З ІНТЕГРУВАННЯ ЕЛЕМЕНТІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ОСВІТНЮ WEB-СИСТЕМУ

У сучасному світі в останні десятиріччя широкого поширення набули проблеми якісного навчання, розробка нових освітніх методів, адже інформацію, яку потрібно вивчити людині, вона повинна вивчити швидко, але дуже добре запам’ятати і подалі мати можливість користуватися нею. Тому у навчальний процес прийшли новітні технології, які дають змогу більш глибоко зануритися у освітній процес, більш детально вивчити матеріал.

Дуже важливим пунктом в освітньому процесі є практичні зайняття – без них усі знання нічого не коштують. Вони дають змогу не тільки закріпити знання, а й показати поведінку у реальному житті.

Важливою проблемою, яку вирішують за допомогою віртуальної реальності в освіті, є надання можливості для практичних зайнять, недоступні у реальному житті через небезпеку або брак коштів. Віртуальна реальність дає змогу пробувати, тестувати набуті знання у віртуальному світі, який майже нічим не відрізняється від реального.

Але сама по собі віртуальна реальність не допоможе покращити якість освітнього процесу, тому потрібен спеціальний підхід, який буде вирішувати наступні проблеми:

- Обмеження кількості та наповнення завдань, які можна представити у віртуальній реальності.
- Зручність використання застосунку.
- Наближення системи до реальних практичних зайнять.

Тому було запропоновано дослідити методи інтеграції віртуальної реальності та можливість їх застосування для вирішення вищезгаданих проблем.

Система інтеграції елементів віртуальної реальності буде реалізована на базі

освітньої WEB-системи, яка моделює процеси розумного будинку.

Для вирішення проблеми обмеження кількості та наповнення задач у застосунку, потрібно розробити інтерфейс, який би дозволяв контролювати контент та наповнення завданнями додаток через саму освітню систему, а також розробити архітектуру системи для двостороннього зв’язку освітньої системи і саме застосунку.

Як сховище кодової бази буде використано централізовану систему контролю версій Git та обгортку для неї GitHub.

Для вирішення проблеми зручності взаємодії та використання застосунку користувачем, потрібно розробити зручний та зрозумілий інтерфейс взаємодії із віртуальною реальністю.

Проблему наближення системи до реальних практичних завдань потрібно вирішити за допомогою реалізації реалістичної графічної сторони застосунку, за допомогою симуляції реальних процесів розумних будинків у застосунку, а також за допомогою такого інтерфейсу взаємодії користувача із застосунком, щоб створювалась ілюзія взаємодії із справжнім світом.

Всі вищезазначені операції будуть виконані виключно за перевіреними досвідом міжнародними стандартами з використанням найкращих практик розробки програмного забезпечення.

2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ ДЛЯ ЗАДАЧІ

Після вивчення ринку технологій і пошуку схожого програмного забезпечення для інтеграції елементів віртуальної реальності в освітній процес було знайдено три системи, які дають можливість інтегрувати віртуальну реальність в освіту: ClassVR, Unimersiv та ENGAGE.

2.1. Система ClassVR

Однією із розглянутих систем із інтеграції елементів віртуальної реальності в освітній процес є система ClassVR.

Система ClassVR – це система із інтеграції віртуальної реальності в освітній процес шляхом розробки власних курсів для навчання. Особливістю цієї системи є те, що вона використовує власні окуляри віртуальної реальності, мають власну бібліотеку курсів, які були розроблені спеціалістами із співпрацею з вчителями американських шкіл, розроблений спеціальний інтерфейс взаємодії із віртуальною реальністю як для школярів, так і для вчителів, власні станції обслуговування окулярів, а також систему доповнення та додавання контенту до сцен у віртуальній реальності.

Окуляри віртуальної реальності є точкою входу саме у неї, тому дуже важливо, щоб їх було зручно використовувати, тому у системі ClassVR розробили власні окуляри, які є абсолютно автономними, тобто не потребують підключення до комп’ютера за допомогою кабелю. Вони мають стильний зовнішній вигляд для того, щоб привертати увагу школярів, мають зручні ремені для зручної фіксації та регулювання на голові. Самі окуляри та ремені зображені на рисунку 2.1.

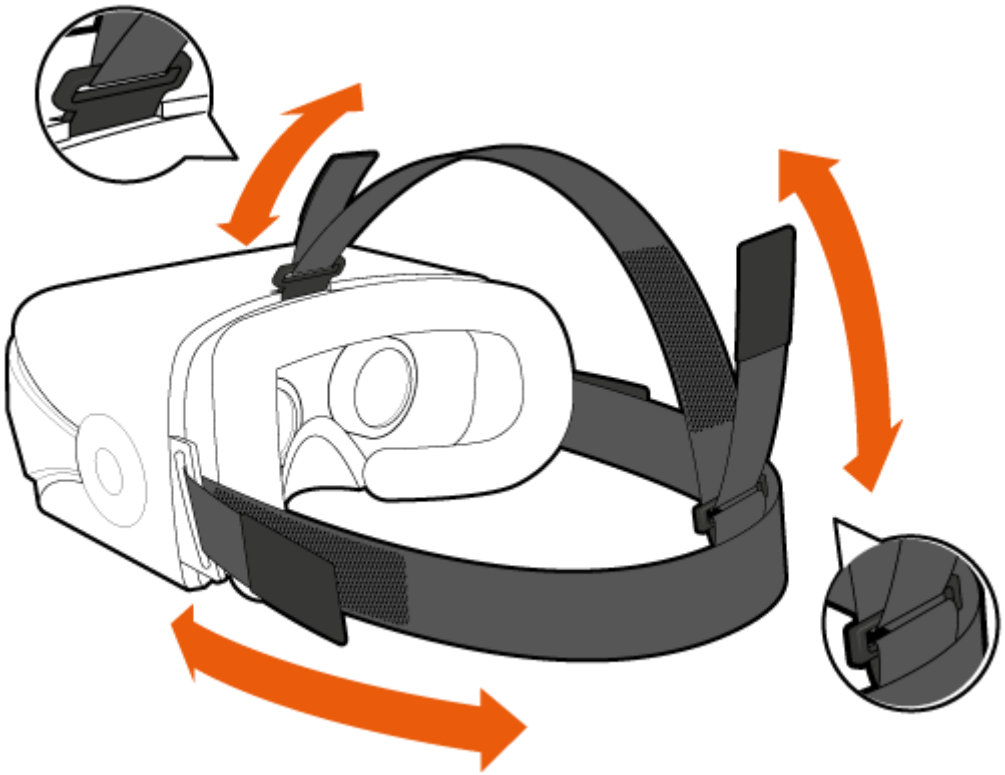

Рисунок 2.1 Зовнішній вигляд окулярів та ременів ClassVR

Ці окуляри мають зручний форм-фактор, м’які підкладки, щоб людина будь-якого віку могла використовувати ці окуляри впродовж тривалого часу.

Також у системі ClassVR реалізовано зручний та зрозумілий інтерфейс користувача, який добре підходить до шкільної тематики і подобається учням. Інтерфейс є кольоровим, дуже простим для використання, дозволяє швидко та легко переключатися між завданнями та сценами не втрачаючи фокусу на поточному завданні.

У ClassVR використано особливу систему взаємодії користувача із віртуальною реальністю: окуляри мають спеціальну камеру, яка бачить рухи руками користувача. За допомогою цієї системи, окуляри та додаток мають змогу бачити жести користувача і реагувати на них належним чином. Тому взаємодія із віртуальною реальністю відбувається без інших пристроїв або контролерів, тільки за допомогою жестів. Це зображено на рисунку 2.2.

Але без інтерфейсу управління система ClassVR була би просто набором сцен та завдань, які не можна ніяк регулювати. Тому однією складовою цієї системи є інтерфейс саме вчителя. Він представляє собою WEB-застосунок, який дає змогу

вибирати із більш ніж 700 курсів потрібний. Також за допомогою цього інтерфейсу вчитель має змогу вибирати, які завдання та сцени будуть показуватися саме поточній групі студентів, може ставити їх у власний порядок, або ж ставити окремому студентові окремі завдання.

Також у цій системи є можливість додавати свої сцени. Це можуть бути 3D об’єкти, відео 360 або фотографії. Вчитель може завантажувати їх на власних розсуд у систему, та ставити у чергу як і інші завдання чи сцени.

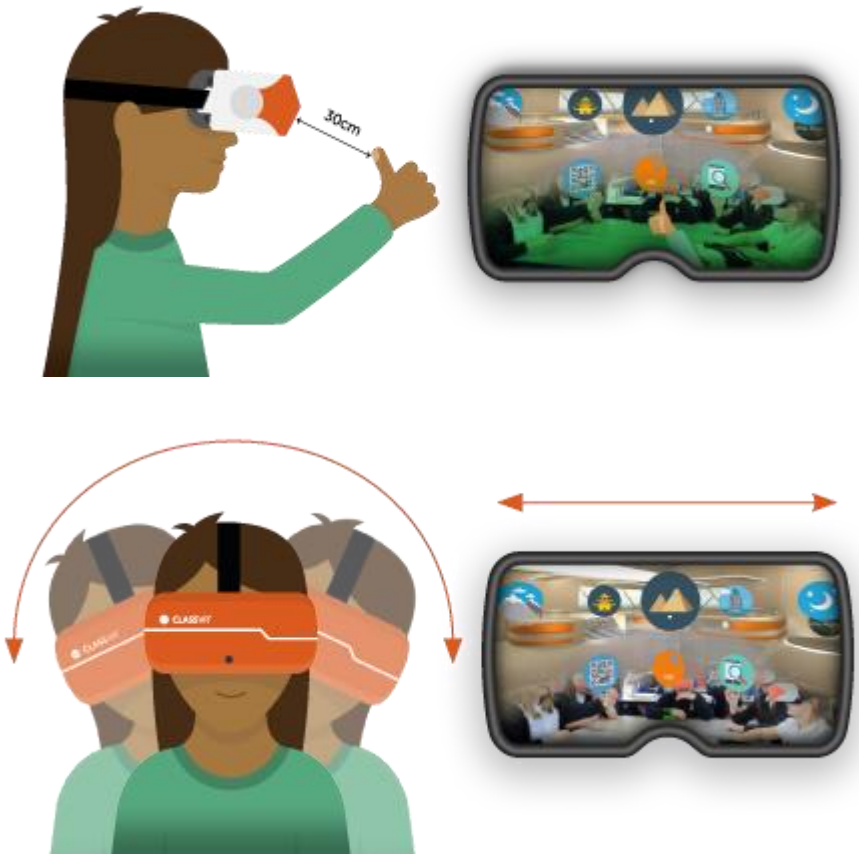

Рисунок 2.2 Система жестів у ClassVR

Для того, щоб мати змогу використовувати цю систему, потрібно заключити контракт та придбати ліцензію на використання цієї системи. Вона вже показала свою спроможність покращувати та робити навчальний процес більш привабливим для дітей, а також показує, що віртуальна реальність справді може покращити рівень сприйняття нових знань учнями.

2.2. Система Unimersiv

Іншою системою для реалізації інтеграції елементів віртуальної реальності була система Unimersiv.

Система Unimersiv – це застосунок для пристроїв Oculus Rift та Samsung Gear, який надає доступ до різних сцен у віртуальній реальності для тренування людей у різних сферах.

Це комерційна платформа, у якій можна замовити різноманітні сцени у віртуальній реальності для тренування людей у будь-якій сфері, або получити доступ до вже існуючих сцен.

Головною особливістю цієї платформи є те, що завдання та сцени націлені більше на практичні завдання, як от поведінка на космічній станції або практичні зайняття з керуванням вантажопідйомника.

Також в системі Unimersiv використовується реалістична графіка, яка доступна через використання потужних окулярів віртуальної реальності Oculus Rift, приклад графіки наведено на рисунку 2.3. Це можливе також через те, що основною аудиторією даної системи є робітники різноманітних підприємств та закладів, тому стає можливим використання більш складного інтерфейсу.

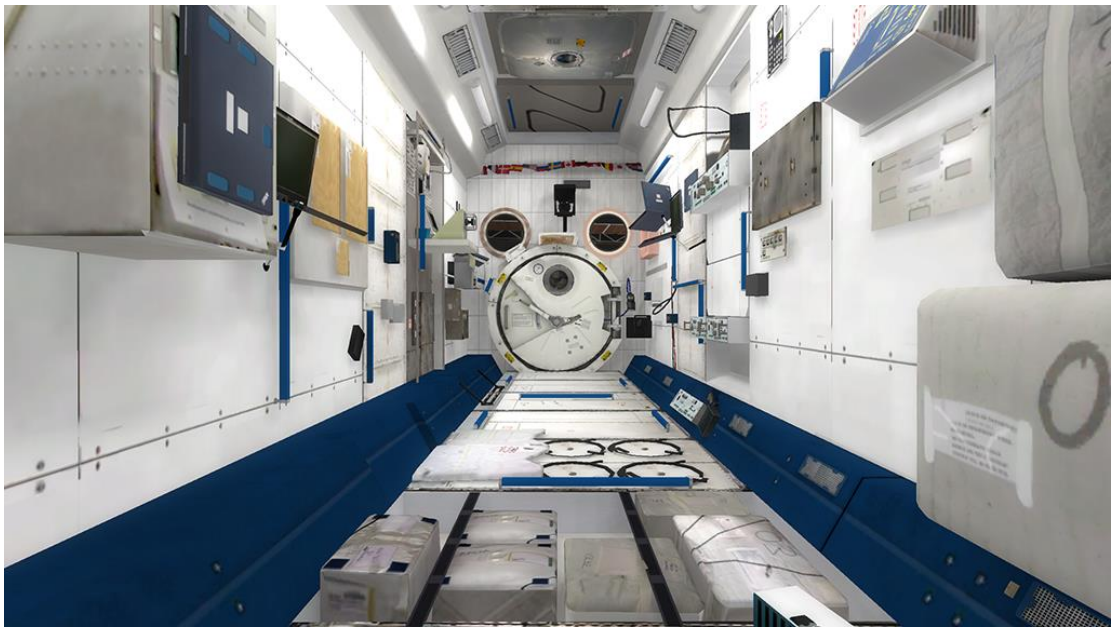

Рисунок 2.3 Приклад графіки у системі Unimersiv

Система Unimersiv дає можливість використати віртуальну реальність як дуже реалістичний тренажер для різноманітніших практичних завдань. Все, що потрібно, щоб почати, це встановити спеціальний додаток, замовити доступ до сцени і починати практичні зайняття.

2.3. Система ENGAGE

Останньою розглянутою системою для реалізації системи інтеграції елементів віртуальної реальності в освітній процес була система ENGAGE.

Система ENGAGE – платформа для соціального навчання, презентації і корпоративного навчання. Вона дозволяє вчителям та корпораціям створювати зустрічі, класи, приватні уроки і презентації з людьми по всьому світу у безпечному, віртуальному, багатокористувацькому просторі.

Віртуальні об’єкти і віртуальний простір системи ENGAGE дозволяє створювати спеціалізовані уроки, які покращують вивчення матеріалу і взаємодію під час уроку.

Особливістю цієї платформи є те, що користувач може записати свої уроки, виступи, презентації та інші навчальні матеріали та поширювати їх не тільки для усіх користувачів платформи, а робити їх доступними тільки окремим користувачам. Тобто можна створювати навчальні матеріали тільки своїй групі людей, а інші їх не будуть бачити.

Створення уроків проходить за допомогою спеціального інтерфейсу, в якому можна завантажувати відео, картинки, 3D моделі. Вже всередині застосунку у віртуальній реальності можна побудувати власну сцену, можна задати порядок показу картинок або відео.

Для того, щоб створити симуляцію саме навчального процесу, у системі ENGAGE було використано систему аватарів, тобто кожна людина має свій віртуальний вигляд, які бачать інші користувачі. На рисунку 2.4 показано симуляцію уроку в університеті для людей, які фізично не можуть бути присутніми на лекції, але

за допомогою системи ENGAGE вони мають змогу отримати знання від лектора, який наглядно у вигляді лекції розказує матеріал.

Рисунок 2.4 Лекція у системі ENGAGE

Система ENGAGE використовує віртуальний планшет як інтерфейс взаємодії користувача із системою, в якому вчитель може налаштовувати сцени, зайняття та завдання, а учень – записуватися на зайняття, запитувати вчителів та налаштовувати свій аватар.

2.4. Висновки до розділу

Таким чином було розглянуто три основних й найпопулярніших системи для інтеграції елементів віртуальної реальності в освітній процес.

Кожна з них має свої переваги та недоліки.

Система ClassVR має багато різноманітних завдань, зручні окуляри віртуальної реальності, можливість виставляти чергу завданням та контролювати те, що будуть бачити учні із одного пристрою. Також ця система має систему жестів, що дає

можливість не використовувати інші контролери для взаємодії із віртуальною реальністю. Однак все одно окуляри потрібно придбати, нічого, окрім цієї системи туди не можна завантажити. Також увесь контент розрахований для дітей, тому не мають місце складні симуляції.

Другою розглянутою системою стала Unimersiv. Ця система дозволяє проходити практичні завдання із реалістичною графікою та системою взаємодії. Але ця система не може бути розширеною користувачем – потрібно звертатися до розробників для нової симуляції. Також усі практичні зайняття жорстко обмежені.

І останньою розглянутою системою стала ENGAGE. Ця система розроблена спеціально для колективного навчання, симуляції презентацій та конференцій для людей зі всього світу. Але дуже маленька можливість додавати нові зайняття, вони обмежені лише відео, картинками та 3D моделями, які можна лише розташувати у просторі.

Тому питання доцільності розробки системи інтеграції елементів віртуальної реальності в освітню систему, яка буде реалізовувати всі переваги та не включати всі недоліки вище перерахованих систем, стає ще більш пріоритетним.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ІНТЕГРАЦІЇ ЕЛЕМЕНТІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ОСВІТНЮ СИСТЕМУ

Одним із найважливіших завдань при розробці програмних продуктів є вибір таких засобів, які б полегшили роботу програміста, надавши всі необхідні інструменти для реалізації поставленого завдання, і дали б змогу отримати результат, який повністю задовольняє користувача.

Для реалізації системи інтеграції елементів віртуальної реальності в освітню систему було використано технологію відображення віртуальної реальності Cardboard, ігровий двигун Unity3D та мову C# як основну мову написання коду та ігровий багатофункціональний контролер MadCatz.

3.1. Відображення віртуальної реальності

Віртуальна реальність – це симуляція справжнього світу у віртуальному просторі. Але для того, щоб користувачу дійсно здавалося, що він знаходиться в іншій реальності, потрібно, щоб його органи відчуття приймали сигнали саме від віртуального простору. Більшість інформації людина сприймає за допомогою очей, тому саме заміна зображення реального світу віртуальною картинкою і є найбільшою задачею для розробників застосунків із використанням технологій віртуальної реальності.

На сьогоднішній день єдиним прийнятим стандартом відображення віртуальної реальності є окуляри віртуальної реальності [1]. Вони бувають різні та мають різну конструкцію. Але найбільш розповсюдженою є конструкція із двома лінзами, кожне для окремого ока людини, через які вона дивиться на екран, який поділений на дві частини, для кожної лінзи окремо. Приклад внутрішнього розташування елементів

можна побачити на рисунку 3.1.

Рисунок 3.1 Внутрішнє розташування елементів у окулярах

Завдання зіниць в наших очах полягає в тому, щоб змінити потік вхідного світла таким чином, що він фокусувався на рецептори на задній частині очей. Зіниця вигинається в залежності від відстані між очима і від того, на що людина дивиться. Якщо дивитися на щось дуже близько, то зіниці повинні дуже сильно вигнутися, щоб людина побачила чітке зображення. Якщо дивитися на щось на відстані, зіницям не потрібно сильно вигинатися.

З віком, зіниці стають менш здатними до вигинання і змінювання потоку вхідного світла, тому підлітки можуть зосередитися на речах на відстані близько 7 см перед їхніми очима, а вже літні люди не можуть.

Таким чином люди відчувають труднощі з переглядом віртуальної реальності на спеціальному екрані (VR HMD), який знаходяться перед очима на відстані від 3 до 7 см. Тому нам потрібні лінзи у цьому екрані, які вигинають світло та полегшують завдання оку, щоб побачити чітке зображення. Зазвичай у окулярах віртуальної реальності використовують лінзи Френеля, приклад яких можна побачити на рисунку 3.2.

Рисунок 3.2 Лінза Френеля

Лінзи Френеля мають кільця на своїй поверхні – це дозволяє зменшити кількість матеріалу для її виготовлення при таких же параметрах вигинання світла [2]. Тому саме вони встановлюються у окуляри віртуальної реальності.

Найважливішим елементом окулярів є екран. Зазвичай він ділить усе відображуване зображення на дві частини. Але бувають окуляри, в яких екран відображує тільки одне зображення, але на відміну від двох зображень, в одному не можна досягти такого рівня стереоскопічного зображення, який би зміг обдурити мозок людини. Тому найчастіше використовуються саме екрани із відображенням двох зображень одночасно.

Таким чином можна створити стереоскопічний ефект та обдурити мозок людини, але виникає проблема із тим, що замість прорахування лише одного зображення, пристрій повинен обрахувати одразу два зображення, причому робити це швидко, щоб досягти високого рівня плавності, щоб користувачу не стало зле.

Саме тому довгий час віртуальна реальність була недосяжна, але з розвитком технологій, стало можливим різноманітні форм фактори пристроїв для відображення віртуальної реальності.

3.2. Cardboard

Із розвитком технологій, людям відкривалися все нові способи відображення зображення на екрані в окулярах віртуальної реальності. Окуляри віртуальної реальності не тільки повинні показувати зображення, але й і слідкувати за поворотами і нахилами голови користувача, ці дані потрібно передавати на пристрій, який обчислює та генерує зображення для екрану окулярів. Тому із розвитком саме мобільних телефонів та смартфонів, прийшла ідея замінити зовнішній пристрій, який генерує зображення для окулярів на смартфон.

У 2014 році компанія Google представила свою розробку – Cardboard. Ця розробка – це не стільки концепт окулярів, скільки підхід до створення та відтворення додатків у віртуальній реальності. Самі окуляри виступають лише стереоскопом, тобто мають лише лінзи, ремені та саме корпус. А вже смартфон виступає як і екраном, так і пристроєм, який генерує зображення, так і акселерометром, гіроскопом і компасом для відстежування дій користувача [3]. На рисунку 3.3 зображений концепт мінімалістичних окулярів Cardboard.

Рисунок 3.3 Окуляри Google Cardboard

Така ідея дуже сподобалась людям, адже:

1. Самі окуляри небагато коштують, адже в них зовсім не має електроніки.
2. Смартфон є майже у кожної людини.
3. Для того, щоб запустити додаток, потрібно лише скачати додаток Cardboard із магазину та сам застосунок і просто запустити його і вставити телефон у окуляри.
4. Компанія Google випустила дуже зручну та просту Cardboard SDK для розробників, яка дозволяє дуже швидко та просто створювати додатки із застосуванням технологій віртуальної реальності.

Ця технологія працює досить просто: смартфон виступає обчислювальним пристроєм із датчиками. Симуляція створення двох зображень створюється завдяки розділенню екрану смартфону навпіл та імітації зображення для лінз.

Саме тому було обрано цю технологію для відображення віртуальної реальності.

3.3. Взаємодія із віртуальною реальністю

Окрім відображення зображення віртуальної реальності, користувачу також потрібно взаємодіяти із нею якимось чином. Смартфон може відстежувати повороти головою користувача за допомогою датчиків всередині. Тому це відкриває можливість до найпростішої взаємодії із віртуальною реальністю – це повороти камери всередині додатку, щоб симулювати повороти голови користувача. Це можливе завдяки технології Cardboard, яка всередині за допомогою зчитування даних із датчиків керує поглядом користувача всередині додатку.

Те, що є можливість керувати за поглядом користувача, дає можливість саме поглядом взаємодіяти із віртуальною реальністю. Це є найпростішою формою взаємодії, однак вона має декілька застосувань:

1. Керування інтерфейсом користувача. Коли користувач наводить свій погляд на елемент інтерфейсу, то через деякий час спрацьовує подія взаємодії із цим елементом інтерфейсу. Таким чином можна симулювати взаємодію із будь-

яким елементом інтерфейсу.

2. Взаємодія із об’єктами у віртуальній реальності. Взаємодія із об’єктами можна реалізувати так як із UI, але додаючи ще одну можливість – це можливість переносити об’єкти поглядом.
3. Переміщення користувача. Достатньо подивитися на підлогу деякий час і додаток перемістить користувача на точку погляду.

Але керування поглядом не завжди зручне, коли потрібно прокрутити список чи обрати із декількох варіантів не відводячи погляду. Тому було вирішено використовувати зовнішній контролер для смартфонів. Такі контролери зазвичай підключаються до телефону через Bluetooth. Зазвичай контролер має джойстики для керування переміщенням у додатку, кнопками для взаємодії із навколишнім середовищем, користувацьким інтерфейсом. Було обрано використовувати контролер MadCatz, бо він має зручний форм-фактор та добре з’єднання із телефоном за допомогою Bluetooth 4.0.

Рисунок 3.4 Контролер Mad Catz

3.4. Технології розробки

Для реалізації обрані наступні технології розробки, що дозволяють повністю реалізувати процес поставленої технічної задачі з створення системи інтеграції елементів віртуальної реальності в освітню систему.

3.4.1. Ігровий двигун Unity3D

Двигун Unity3D - це крос-платформний ігровий движок, розроблений компанією Unity Technologies, вперше оголошений і випущений у червні 2005 року на Apple Inc. Починаючи з 2018 року, двигун був розширений для підтримки більш ніж 25 платформами. Двигун можна використовувати для створення тривимірних, двовимірних, ігор, а також застосунків у віртуальній та доповненій реальності. Двигун також використовується галузями за межами відеоігор, таких як фільми, автомобільна, архітектура, інженерія та будівництво.

Двигун Unity надає користувачам можливість створювати ігри та додатки як в 2D, так і в 3D, а движок пропонує основний API для написання сценаріїв використовуючи мову програмування C#, як для редактора Unity у вигляді плагінів, так і для самих ігор [4]. До того, як C# була основною мовою програмування, яка використовувалася для двигуна, він раніше підтримував Boo, який був видалений з випуском Unity 5, і версію JavaScript під назвою UnityScript, яка була видалена у серпні 2017 року, після виходу Unity 2017.1, на користь C#.

У 2D-іграх Unity дозволяє імпортувати спрайти і робити розширений 2D-рендерінг світу. Для 3D-ігор Unity дозволяє специфікувати стискання текстур, і роздільну здатність для кожної платформи, яку підтримує ігровий двигун, і надає підтримку для відображення рельєфу, відображення паралакса, дозволяє робити рендер тіней, світла

за допомогою мап світла та тіней.

Станом на 2018 рік, Unity використовувалася для створення приблизно половини мобільних ігор на ринку і 60% усіх додатків, які використовують доповнену та віртуальну реальність, тому журнал Fortune сказав, що Unity "домінує над бізнесом віртуальної реальності".

Двигун Unity3D дозволяє дуже швидко та просто створювати 3Д додатки за допомогою системи сцен, приклад на рисунку 3.5. Кожна сцена являє собою набір 3Д чи 2Д об’єктів, які мають основні властивості, такі як позиція в світі, поворот та розмір – ці параметри зберігаються в компоненті Transform, яка є обов’язковою для всіх об’єктів. Кожен об’єкт має назву та може мати дочірні об’єкти.

Рисунок 3.5 Приклад сцені у Unity3D

Також двигун надає можливість додавати різноманітні компоненти до об’єктів, змінюючи їх вигляд, поведінку та багато інший властивостей. Декілька компонентів, які можна добавити до об’єкта:

1. Collider – додає фізику колізій до об’єкта, цей компонент починає рахуватися як стінка, об яку будуть битися інші об’єкти із цим компонентом.
2. Rigidbody – додає фізику до об’єкта. На нього починають діяти сили, наприклад

сила тяжіння.

3. Camera – компонент камери, все, що попадає у її поле зору попадає до зображення на екрані.
4. MeshFilter та MeshRenderer – компоненти, які присутні на кожному 3Д об’єкті, який має меш. Вони показують його згідно того, який матеріал доданий до цих компонентів.
5. UI компоненти – це компоненти, які дозволяють створювати та відображати користувацький інтерфейс.
6. Скріпти – користувацький компонент, який створюється написанням коду на C#. Цей компонент є по суті класом, який є нащадком класу MonoBehaviour. Він забезпечує компонент деякими методами, такими як: Start, Awake, Update, LateUpdate, OnDestroy, OnAwake. Це дає можливість створювати власну поведінку об’єкта на різних стадіях його життя.

Важливою особливістю Unity – це те, що він дозволяє створювати матеріали не тільки для 3Д моделей, а й для 2Д і користувацького інтерфейсу. Матеріал – це сутність, на яку можна додати шейдер. Кожен матеріал – це представляє собою шейдер із заданими налаштуваннями [5].

Цей двигун дозволяє створювати свої шейдера. Шейдер – це маленька програма, яка виконується на відеокарті. Вони можуть застосовуватися для різних цілей: як для відображення текстур, ефектів, так і зміні вершин 3Д моделі, так і для генерації світла і пост обробки зображення [6]. Unity версії 2018 і новіше дає можливість використовувати Shader Graph, який дозволяє не писати код шейдеру на спеціалізованій мові, а використовувати різноманітні операції для створення вихідного зображення, приклад на рисунку 3.6.

Двигун Unity3D дає можливість створювати власний конвеєр рендерингу зображення (LDRP). Він дозволяє змінювати методи візуалізації світла, додавати різні ефекти [6]. Також його функцією є створення чіткого зображення за найменших витрат потужності пристрою.

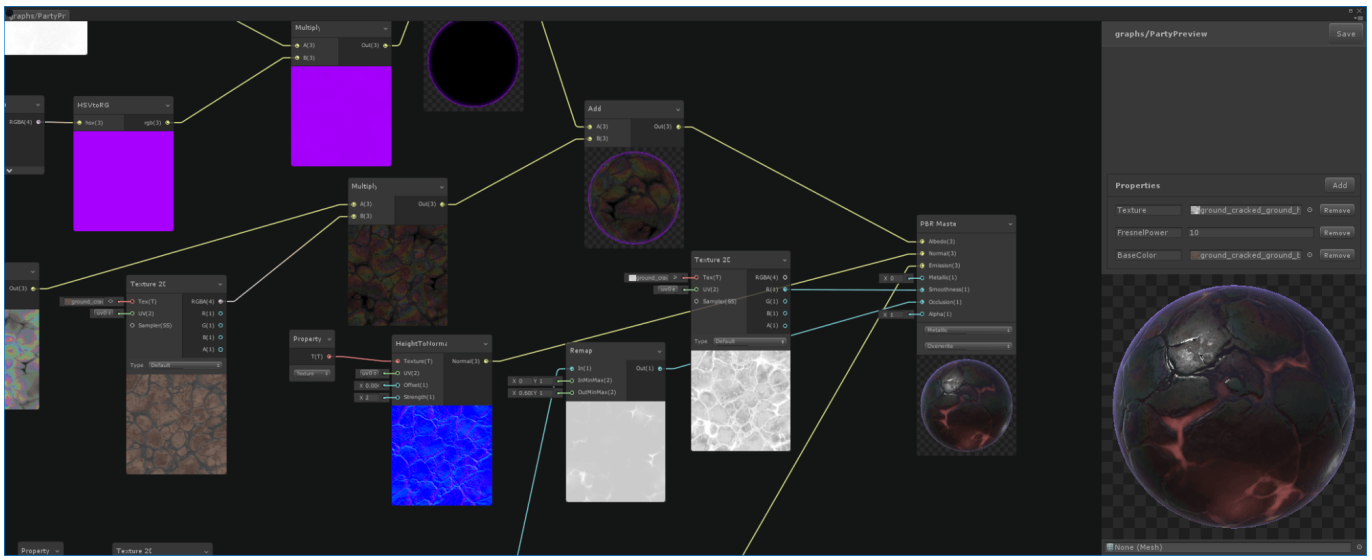

Рисунок 3.6 Shader Graph

Також Unity дає можливість додавати пост обробку зображенню і додавати різноманітні ефекти до нього. Такими ефектами є:

1. Згладжування (anti-aliasing) - технологія, що використовується в обробці зображень з метою зробити межі кривих ліній більше гладкими, прибираючи «зубці», що виникають на краях об'єктів [8].
2. Motion blur – ефект, який робить зображення не чітким при швидкому повороті камери.
3. SSAO - імітує розсіяне непряме освітлення і відповідне затемнення в тривимірному віртуальному просторі.

Тож Unity3D дає можливість створювати додатки у 3D та 2D та у віртуальній реальності із використанням багатьох інструментів, ефектів та додавати фізику, світло та тіні до об’єктів, що забезпечує повним набором для побудови додатків.

3.4.2. Мова C#

Мова C# - це мова програмування, яка реалізує декілька парадигм програмування [12]. Вона була розроблена в 2000 році корпорацією Майкрософт в рамках її

ініціативи.

Мова покликана бути простою, сучасною, універсальною, об'єктно-орієнтованою мовою програмування.

Мова і його реалізації повинні забезпечувати підтримку принципів розробки програмного забезпечення, таких як перевірка типу, перевірка меж масивів, виявлення спроб використання неініціалізованих змінних і автоматичне збирання сміття. Надійність, довговічність та продуктивність програмного забезпечення є важливими.

Мова призначена для використання в розробці програмних компонентів, придатних для розгортання в розподілених середовищах [9].

Портативність дуже важлива для вихідних кодів і програмістів, особливо тих, які вже знайомі з C і C ++.

Підтримка інтернаціоналізації дуже важлива.

C # призначений для написання додатків як для розміщених, так і для вбудованих систем, починаючи від дуже великих, які використовують складні операційні системи, аж до дуже малих, що мають спеціальні функції [13].

Незважаючи на те, що C# додатки призначені для економного відносно вимог до пам'яті та потужності обробки, мова не мала на меті конкурувати безпосередньо з продуктивністю та розміром з мовою C або мовою асемблера [11].

Але неможливо розглянути мову C# окремо від Unity3D, якщо використовувати цей двигун. Він дає можливість створювати не тільки власні класи як в звичайному C#, а й створювати класи компоненти.

Головною особливістю таких класів є те, що вони є нащадками класу MonoBehaviour [7]. Цей клас дає можливість використовувати методи, які вже є визначеними:

1. Awake – метод, який викликається один раз за життя об’єкту за його ініціалізації.
2. OnEnable – метод, який викликається кожного разу, коли об’єкт включають.
3. Start – метод, який викликається на початку життя об’єкту, але після попередніх методів, перед першим кадром.

4. Update – метод, який викликається кожного кадру.
5. FixedUpdate – метод, який зазвичай використовується для обрахування фізики об’єкту.
6. LateUpdate – метод, який викликається після методу Update.
7. OnDisable – метод, який викликається при вимиканні об’єкта.
8. OnDestroy – метод, який викликається при знищенні об’єкта.

Таким чином мова C# у зв’язці із Unity3D дає можливість створювати потужні додатки на цьому двигуні, використовуючи усю простоту та могутність мови C#.

3.5. Висновки до розділу

Таким чином обравши окуляри віртуальної реальності для мобільних телефонів та використовуючи технологію Cardboard, стало можливим розробка додатку у віртуальній реальності, який майже кожна людина зможе запустити в себе.

Використання бездротового контролера дозволило вирішити питання взаємодії користувача із віртуальною реальністю.

Також із допомогою ігрового двигуна Unity3D та написанням скриптів на мові C# стала можливою розробка красивого, функціонального додатку на мобільні телефони із використанням технологій віртуальної реальності.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ СИСТЕМИ ІНТЕГРАЦІЇ ЕЛЕМЕНТІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ В ОСВІТНЮ WEB-СИСТЕМУ

Система призначена для двох акторів: користувача, який бажає пройти завдання та сцени у віртуальній реальності, або викладача курсів.

Для того, щоб мати можливість користуватися системою користувач повинен зареєструватися або увійти у систему.

4.1. Структура системи

Програмне забезпечення для розв’язання поставленої задачі спроектовано з дотриманням парадигм об’єктно-орієнтованого програмування [10].

Під час розробки системи була реалізована така схема роботи, рисунок 4.1. Центром всієї системи є сервер, через який спілкуються веб застосунок і мобільний додаток. Усі дані зберігаються у базі даних, які отримуються додатками саме через сервер. Особливістю системи є те, що в обох додатках використовуються абсолютно однакові дані.

Спілкування застосунків відбувається за допомогою HTTP запитів, на які сервер відповідає даними у форматі JSON.

Розглянемо схему роботи розробленого програмного продукту на рисунку 4.2.

Спочатку користувач повинен створити акаунт у системі або увійти у неї. Потім йому потрібно обрати, чи він буде звичайним користувачем, чи хоче створювати курси, лекції.

Рисунок 4.1 Схема роботи системи

Звичайний користувач може дивитися на наявні курси, сортувати їх та обрати курси, які він хоче для їх проходження. Потім він може проходити лекції, завдання, тести та інші види активності, які створить автор курсу. Можна добавити курс у обране, можна купити курс. Можна написати автору курсу та поставити запитання, яке стосується даного курсу.

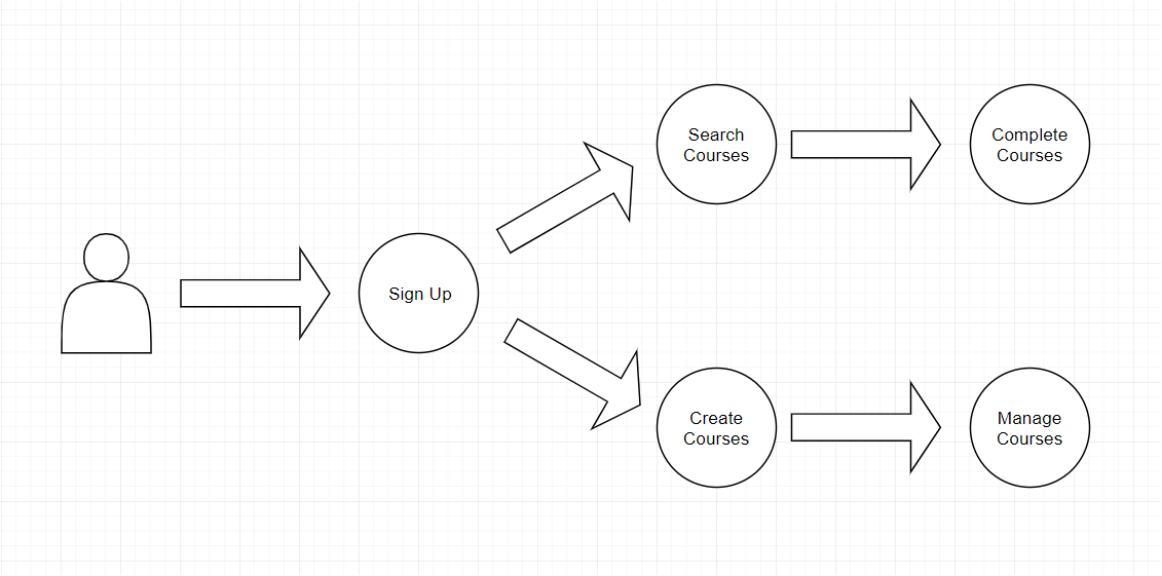

Рисунок 4.2 Схема роботи освітньої системи

Якщо користувач вирішив стати викладачем та автором курсів, то в нього з’являється можливість створення курсів, їх редагувати, створювати лекції та

завдання до курсів, додавати сцени у віртуальній реальності до курсів.

4.2. Інтеграція елементів віртуальної реальності в освітню систему

Для того, щоб інтегрувати елементи віртуальної реальності, було розроблено:

1. Спеціальне API для спілкування сервера і мобільного додатку.
2. Перелік існуючих пристроїв для використання у завданнях із повною інформацією про їхнє призначення, про роз'єми та призначення цих роз'ємів.
3. Створено перелік місць, у які можна встановити пристрої.
4. Спеціальне представлення для зберігання інформації про сцени, завдання та усі кроки, які потрібно зробити для його виконання.
5. Перелік можливих дій користувача, які потрібно зробити для виконання завдання.
6. Розроблений спеціальний інтерфейс для створення сцен та завдань у віртуальній реальності.

Спеціальне API на сервері було розроблено спеціально для спілкування сервера і мобільного додатку. Воно було створене для якомога більшої ефективності та швидкодії передачі даних. Приклад його роботи на рисунку 4.3. Це API включає в себе:

1. Методи GET для стягування інформації про доступні користувачу анімації та завдання. Вони містять в собі id самого завдання та id курсу для того, щоб додаток міг використовувати при подальших запитах. Також вони містять в собі назви курсу та саме уроку.
2. Методи GET для стягування інформації про конкретне завдання чи анімацію через його id. Розроблений формат відповіді у форматі JSON має в собі всі дані про дане завдання із усіма кроками, пристроями, роз'ємами, які

використовуються, місцями, в які потрібно встановити пристрої, а також спеціальні дії для закінчення завдання.

3. Методи POST для того, щоб повідомити сервер про закінчення виконання завдання із подальшим внесенням цієї інформації у базу даних.

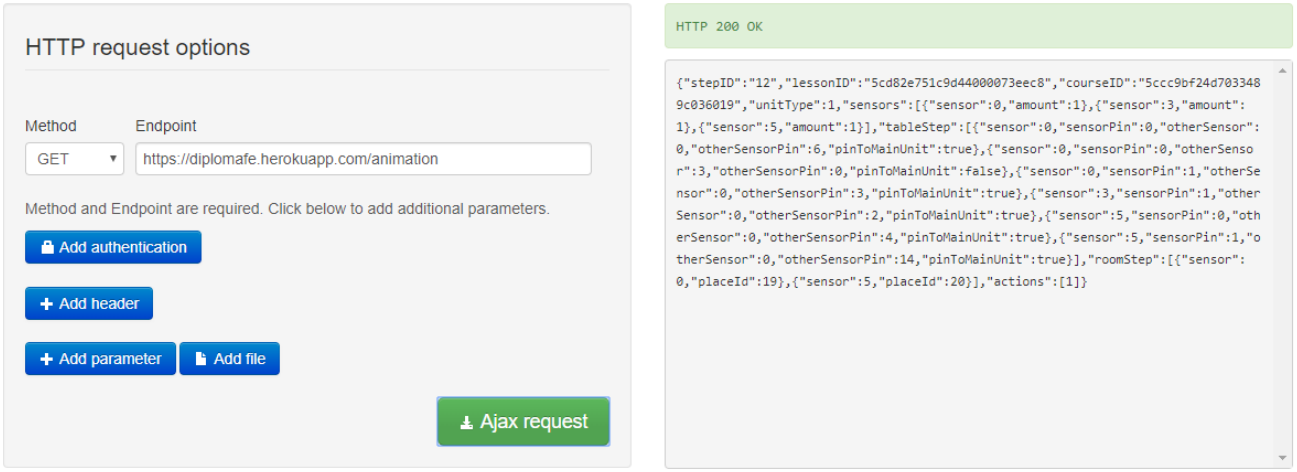

Рисунок 4.3 Приклад роботи API

Перелік існуючих пристроїв для використання у завданнях був розроблений, щоб мати можливість синхронізувати дані про них між сервером та додатком, де самі ці пристрої мають свою поведінку та реакцію на різні дії.

Загалом, було виділено декілька видів пристроїв:

1. Блоки управління.
2. Пристрої для світла.
3. Пристрої для температури та вологості.
4. Пристрої для відслідковування рухів.
5. Пристрої для безпеки.
6. Допоміжні пристрої типу діодів або резисторів.

Усі ці пристрої мають свою назву, поведінку та роз’єми. Для кожного пристрою розроблений перелік усіх його роз’ємів із призначенням кожного. Ці дані також синхронізовані між сервером та мобільним додатком.

Також було створено перелік місць, у які можна розташувати пристрої. Місця для них обиралися згідно того, куди можуть бути розташовані пристрої, які вирішено додати до системи.

Було створено перелік дій користувача, щоб перевірити результат завдання та завершити його. Ці дії задаються у конструкторі завдань і там задається порядок виконування цих дій. У мобільному додатку вже відслідковується виконання цих дій. Після того, як всі вони виконані і користувач перевірив результати своєї роботи, то це слугує сигналом до завершення завдання.

Дуже важливим кроком до інтеграції елементів віртуальної реальності в освітню систему став конструктор саме завдань.

Він дає можливість задати усі пристрої, які потрібні для виконання завдання, задати роз’єми та підключення. Також він дає можливість задати місця для пристроїв та задати ті дії, які потрібно виконати користувачу.

4.3. Опис розроблених алгоритмів

Для розробки даного програмного продукту із інтеграції елементів віртуальної реальності в освітню систему були використані такі технології як Unity3D та Cardboard.

Для реалізації серверної частини використовувалися такі технології як JavaScript, Node.js, Express.js.

Були виконані такі кроки:

1. Встановлений зв’язок із сервером.
2. Отримання від нього даних та серіалізація їх у відповідні типи даних.
3. Визначення спрямованості погляду користувача.
4. Взаємодія користувача із віртуальною реальністю.
5. Проходження завдань.
6. Надсилання результатів до серверу.

Для кращого розуміння продемонстрована діаграма класів на рисунку 4.4. На ній можна поділити систему на чотири основі частини: взаємодія із сервером та серіалізація даних, отриманих від нього, взаємодія користувача із віртуальною

реальністю, керування пристроями та керування графічним інтерфейсом.

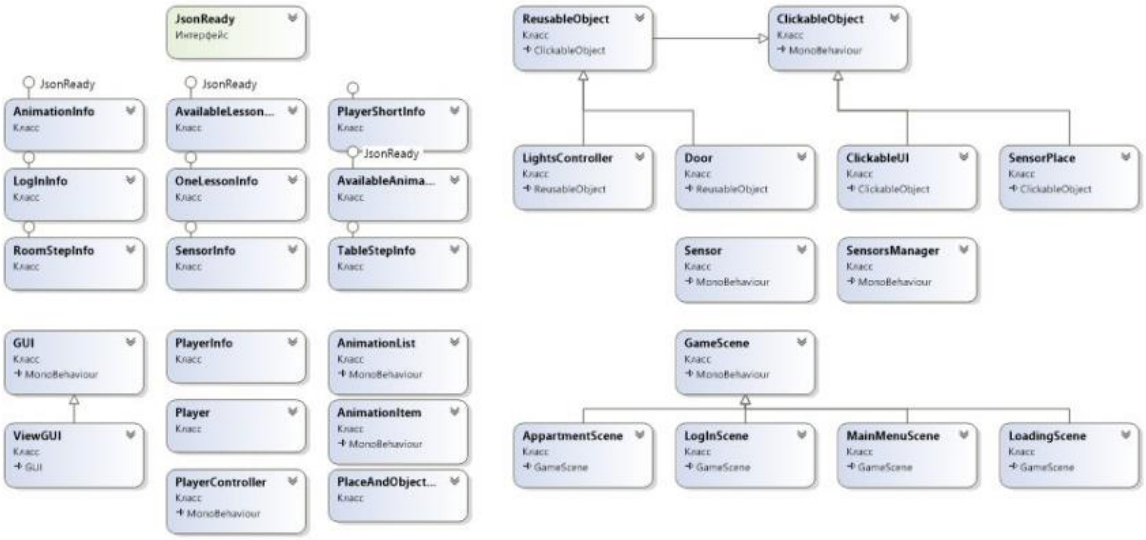

Рисунок 4.4 Діаграма класів програмного продукту

Також продемонстрована діаграма прецедентів, на якій зображені основні дії, що може виконувати користувач, рисунок 4.5.

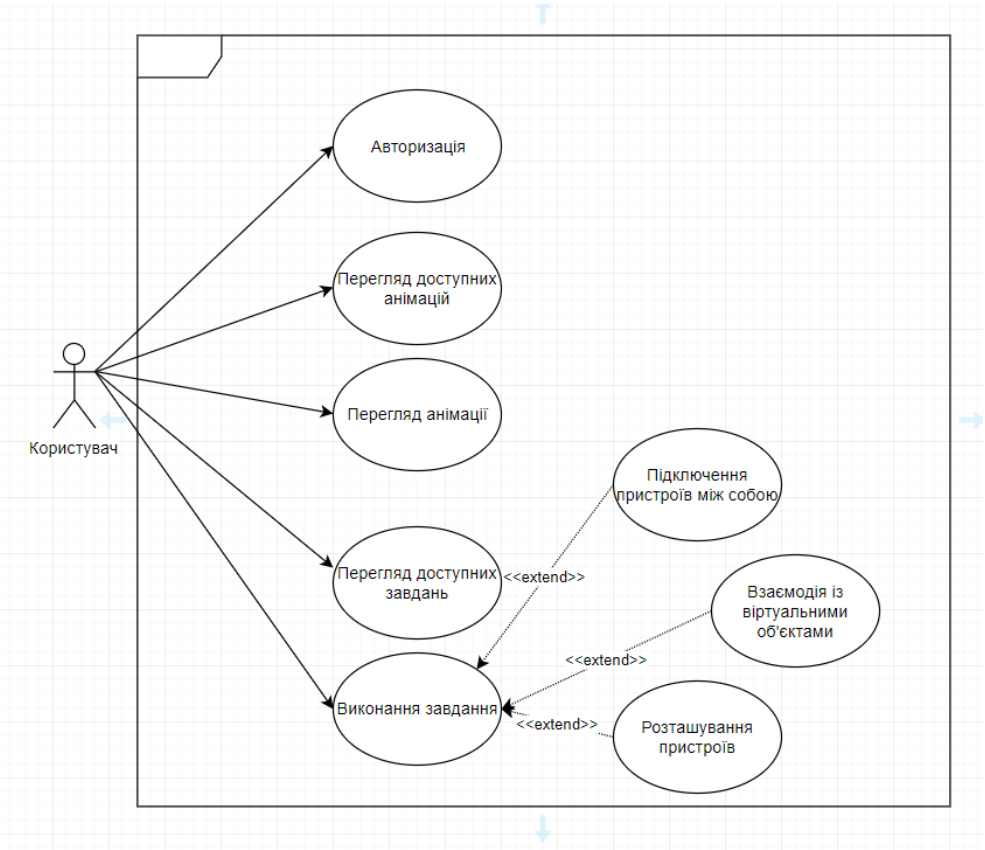

Рисунок 4.5 Діаграма прецедентів системи

4.4. Висновки до розділу

Таким чином, було розроблено систему інтеграції елементів віртуальної реальності в освітню WEB-систему за допомогою декількох способів:

1. Створення API для зв’язку із сервером.
2. Створення типів даних, які однакові для веб-застосунку і для мобільного додатку.
3. Створення інтерфейсу для конструювання сцен та завдань у віртуальній реальності.

Також було створено переліки пристроїв, роз’ємів, місць для пристроїв та перелік дій для перевірки завдань.

Було створено діаграму прецедентів для візуалізації можливих дій користувача.

5. КЕРІВНИЦТВО КОРИСТУВАЧА ІЗ РОБОТИ З СИСТЕМОЮ ІНТЕГРАЦІЇ ЕЛЕМЕНТІВ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ

Щоб запустити додаток, потрібно встановити спеціальний apk файл на смартфон із операційною системою Android та запустити додаток на ньому, підключивши контролер.

5.1. Крок входу в додаток

Для того, щоб скористатися функціоналом системи, потрібно спочатку зареєструватися в системі у веб-додатку або авторизуватися у мобільному додатку, рисунок 5.1.

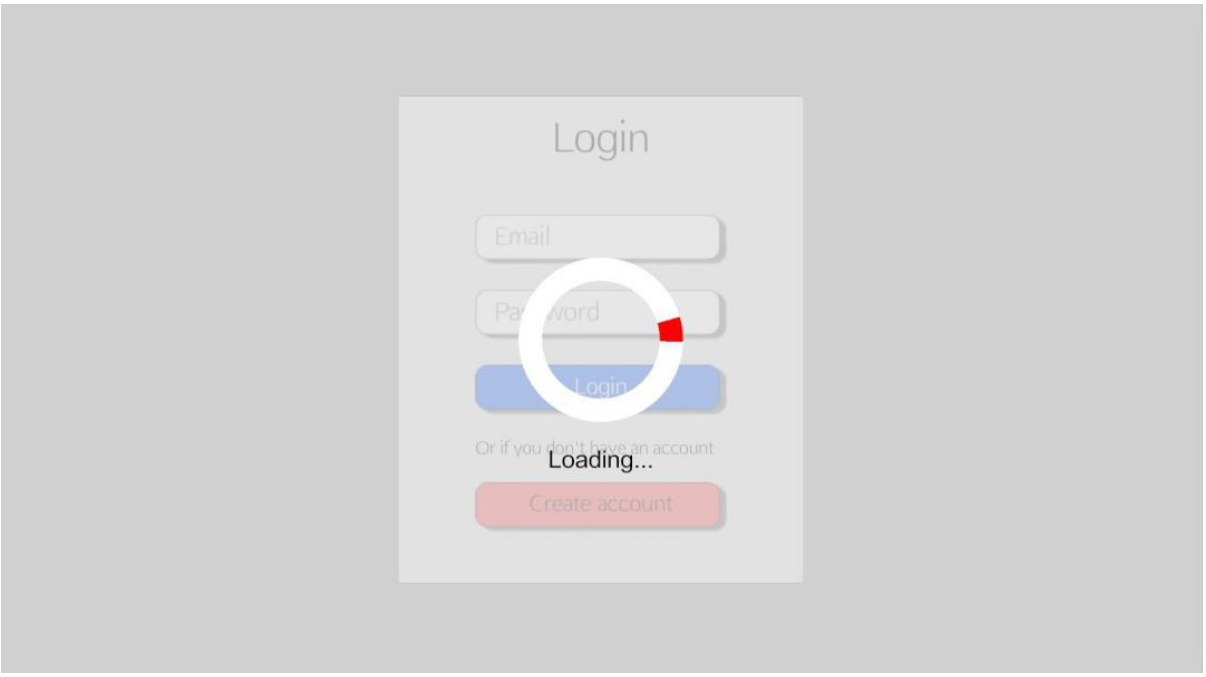

Рисунок 5.1 Форма авторизації під час входу користувача

5.2. Крок вибору завдання

Після входу в систему, користувач попадає на наступну сторінку – головне меню. Звідси він може зробити налаштування додатку, або вибрати на перегляд доступні йому анімації чи завдання, рисунок 5.2.

Після вибору за допомогою погляду та контролера анімацій чи завдань, користувач може побачити список доступних йому завдань чи анімацій із назвами поточного уроку та курсу, рисунок 5.3.

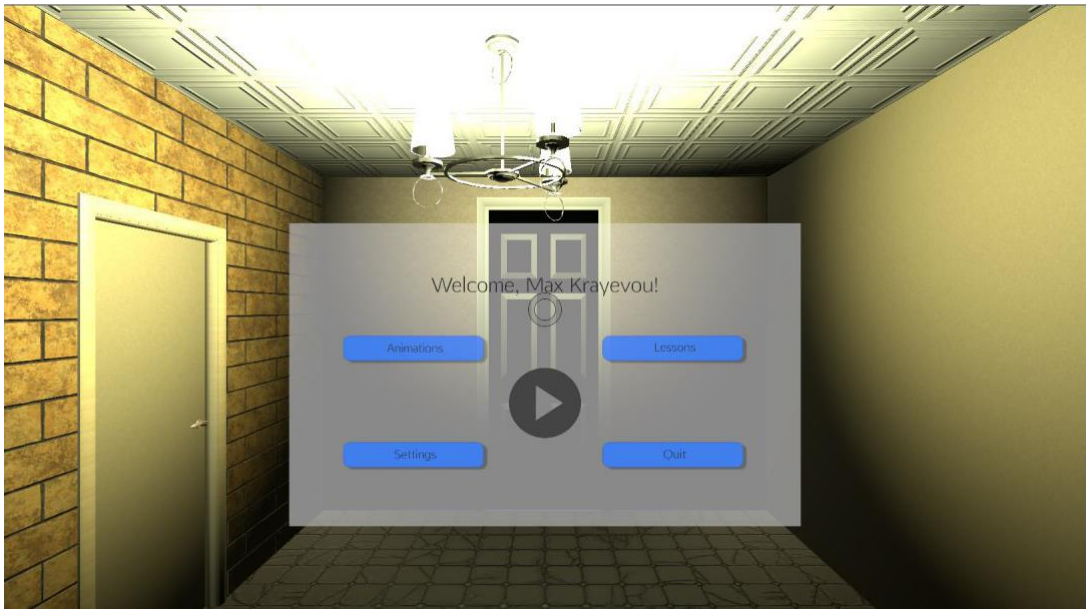

Рисунок 5.2 Головне меню програми

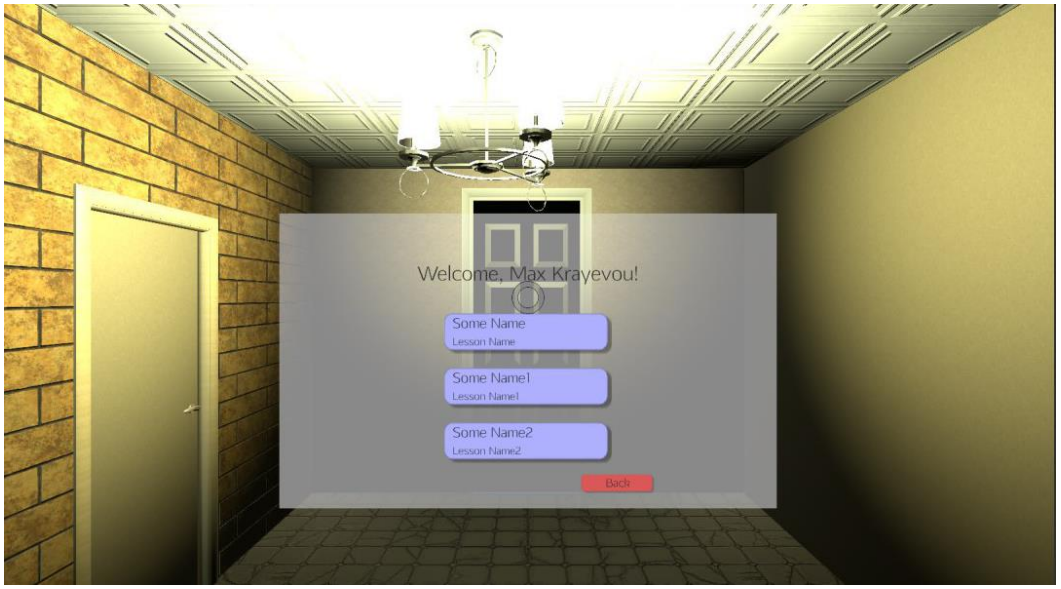

Рисунок 5.3 Перелік доступних завдань для користувача

5.3. Крок з’єднання пристроїв

Після вибору завдання, користувач переноситься у віртуальну сцену, в якій є приклад квартири, у якій він буде проходити завдання.

Першим кроком у завданні завжди є підключення пристроїв один до одного. Користувач може зробити це вибравши із переліку усіх пристроїв той, який він хоче під’єднати до іншого, після цього користувачу показується все роз’єми, які є у двох пристроїв. Далі йому потрібно обрати два роз’єми, які він хоче з’єднати та підтвердити свій вибір, рисунок 5.4.

Рисунок 5.4 Екран з’єднання пристроїв між собою.

Ті пристрої, які вже не мають вільних роз’ємів, помічені написом “Plugged”. Ті роз’єми, які вже використані, помічені написом “Used”.

5.4. Крок установки пристроїв

Після з’єднання усіх пристроїв між собою, користувачу потрібно встановити їх на правильні місця. Для цього йому пропонується вільно ходити по квартирі, взаємодіявши із різними віртуальними об’єктами. Наприклад із дверима, як от на рисунку 5.5 двері закриті і користувачу пропонується відкрити їх, і на рисунку 5.6 вони вже відкриті.

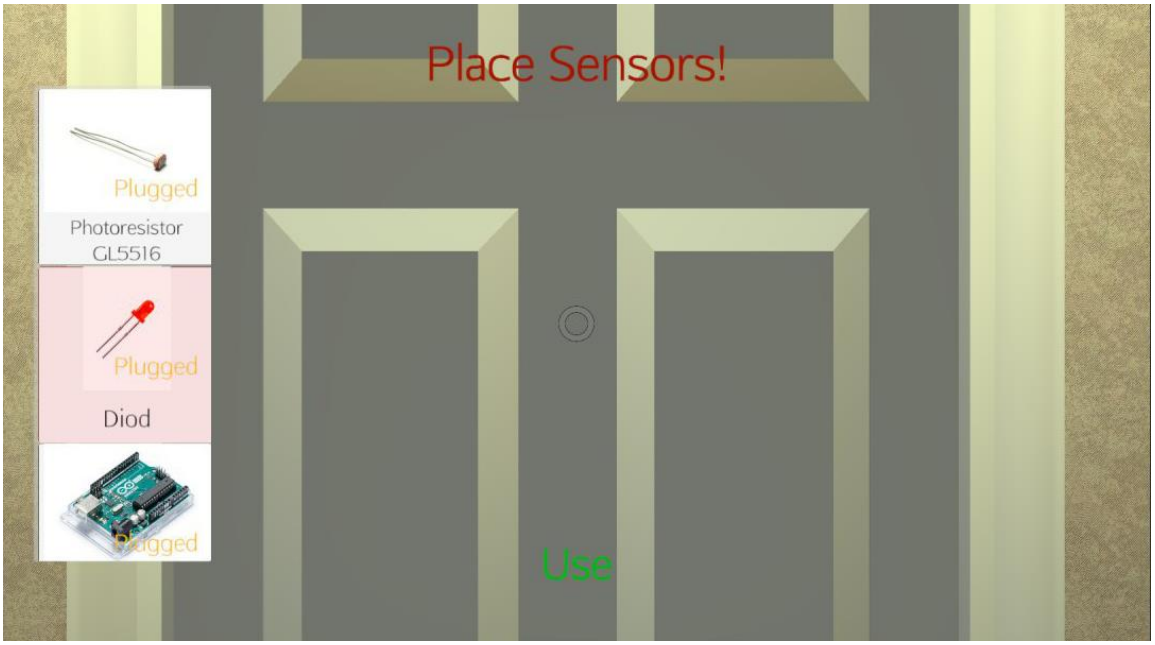

Рисунок 5.5 Користувачу пропонується взаємодіяти із дверима

Рисунок 5.6 Результат взаємодії із дверима

Для того, щоб розташувати потрібні пристрої на місця, йому потрібно їх знайти, тому такі місця підсвічені червоним кольором і користувач може також взаємодіяти із ними, рисунок 5.6.

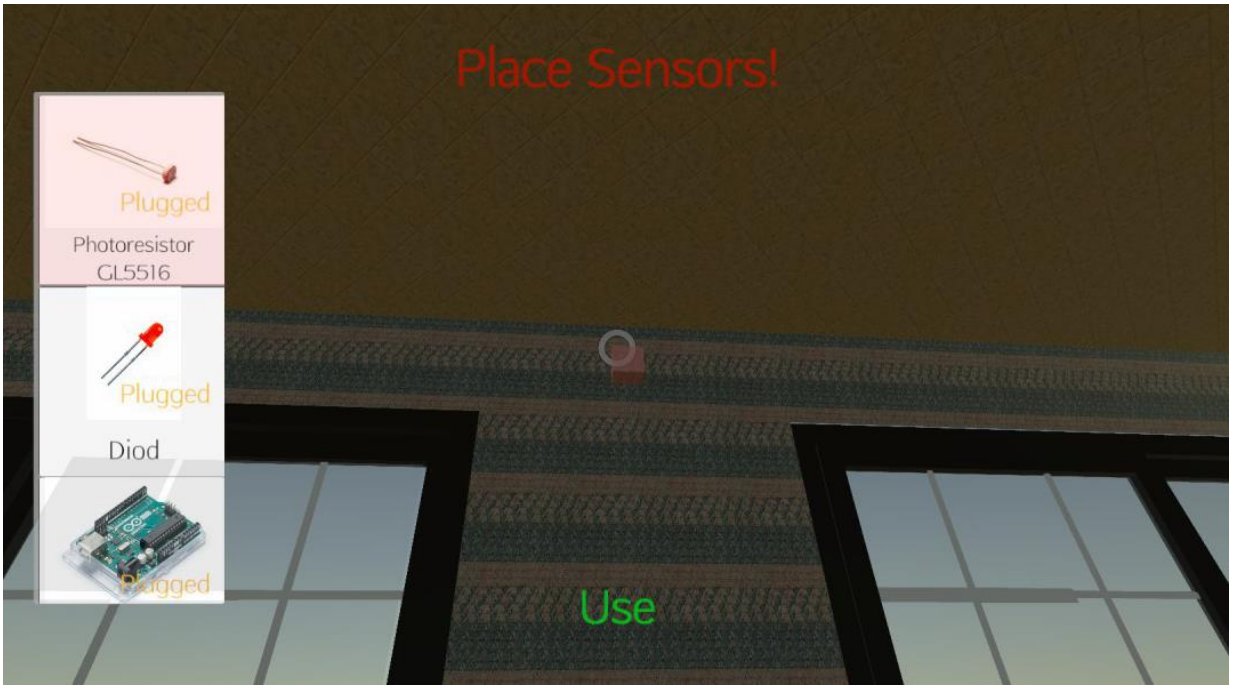

Рисунок 5.6 Місце для встановлення пристрою

Після того, як користувач знайшов правильне місце для встановлення пристрою, йому потрібно вибрати відповідний пристрій із списку і натиснути відповідну кнопку на контролері для встановлення пристрою на місце, рисунок 5.8.

Потрібно розташувати усі пристрої для того, щоб з’явилися дії, які потрібно виконати користувачу для перевірки результату. На рисунку 5.9 можна побачити, що діод, який був встановлений на стіл горить, а зверху екрану є повідомлення про те, що потрібно ввімкнути та вимкнути світло.

Таким чином, після розташування пристроїв по своїм місцям починають діяти зв’язки, які були створені на кроці з’єднання пристроїв. Також деякі пристрої реагують на зовнішнє середовище, як от фоторезістор реагує на вмикання та вимикання світла.

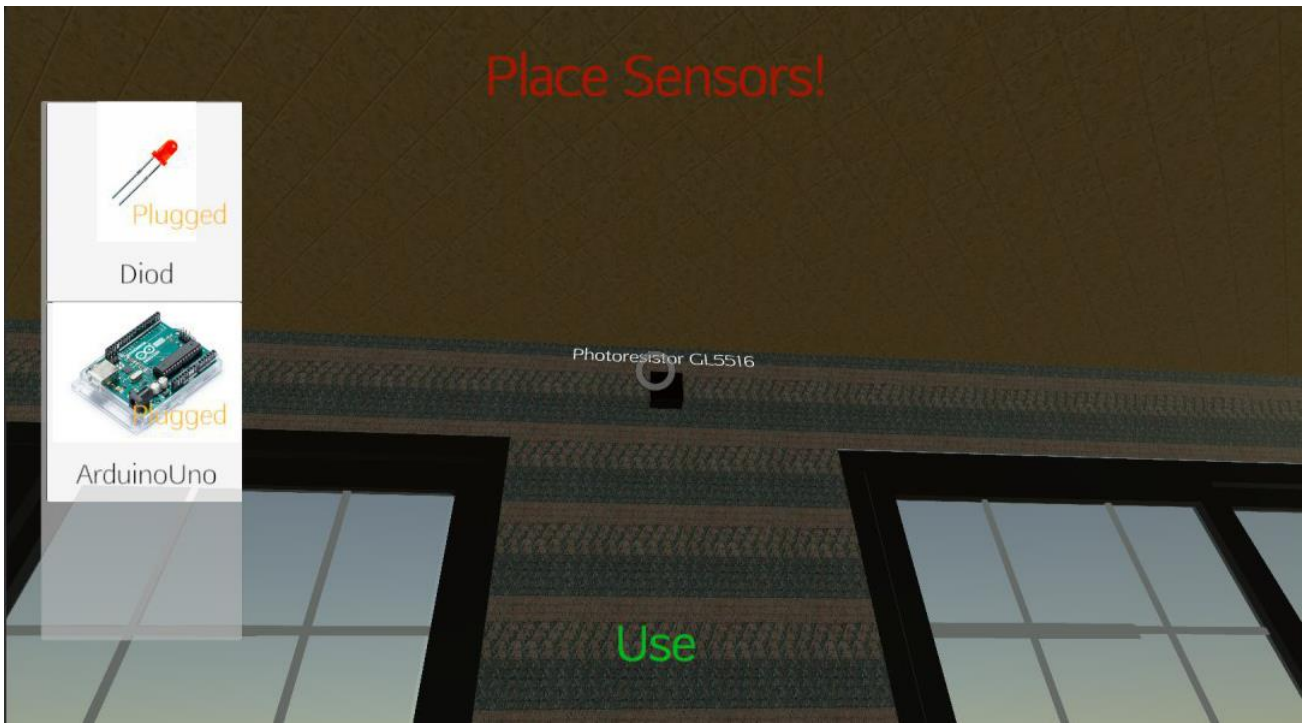

Рисунок 5.8 Встановлення пристрою на місце

Виконання дій потрібно не тільки для того, щоб перевірити результат роботи, а щоб наочно побачити взаємодію пристроїв між собою, із навколишнім середовищем, світлом, температурою, тощо, рисунок 5.10.

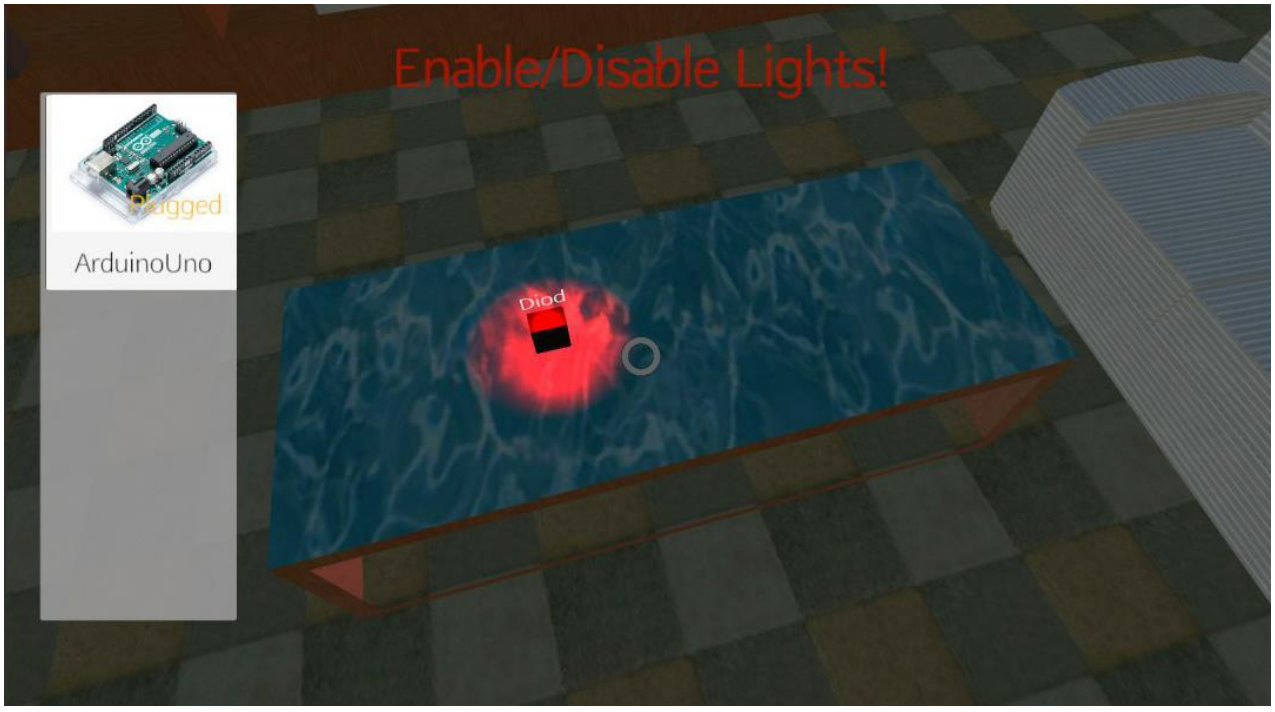

Рисунок 5.9 Поява дій для користувача.

Рисунок 5.10 Візуалізація взаємодії пристроїв із середовищем

Після виконання всіх необхідних дій, користувач може далі знаходитися у завдані та експериментувати або може припинити завдання та потрапити на екран завершення завдання, після якого він потрапляє до головного меню.

Під час завантаження головного меню, додаток відсилає результати виконання завдання до серверу. Якщо до серверу не можна під’єднатися, то застосунок збереже інформацію про виконання завдання у пам’яті, але до вимкнення додатку.

5.5. Висновки до розділу

Таким чином було реалізовано і протестовану систему інтеграції елементів віртуальної реальності в освітню WEB-систему, яка моделює процеси розумних будинків. Вона дозволяє створювати, змінювати та редагувати сцени та завдання у віртуальній реальності, які потім можна пройти у додатку для смартфонів, який виконаний за допомогою ігрового двигуна Unity3D та технології віртуальної реальності Cardboard. Мобільний застосунок тісно інтегрований в освітню WEB-систему за допомогою одного серверу, який працює та віддає однакові дані як веб-

застосунку так і мобільному додатку. Також був реалізований інтерфейс побудови і редагування сцен і завдань у віртуальній реальності у веб-застосунку, розроблене спеціальне API для обміну даними із сервером.

Створено мобільний додаток для смартфонів, у якому за допомогою окулярів віртуальної реальності та спеціального контролера можна вибирати, проходити завдання, які пов’язані із курсами з побудови розумних будинків. Розроблено інтерфейс взаємодії людини із віртуальною реальністю та можливість додавання нових пристроїв до додатку.

ВИСНОВКИ

В процесі виконання роботи було розглянуто технології, які за допомогою який можна побудувати систему інтеграції елементів віртуальної реальності в освітню систему. Були розглянуті переваги та недоліки трьох основних систем для реалізації інтеграції елементів віртуальної реальності в освітній процес, а саме: систему ClassVR, Unimersiv та ENGAGE.

На основі отриманого аналізу була реалізована власна система інтеграції елементів віртуальної реальності в освітню систему, в процесі чого отримано наступні результати:

1. Реалізовано і протестовану систему інтеграції елементів віртуальної реальності в освітню WEB-систему, яка моделює процеси розумних будинків. Вона дозволяє створювати, змінювати та редагувати сцени та завдання у віртуальній реальності, які потім можна пройти у додатку для смартфонів, який виконаний за допомогою ігрового двигуна Unity3D та технології віртуальної реальності Cardboard. Мобільний застосунок тісно інтегрований в освітню WEB-систему за допомогою одного серверу, який працює та віддає однакові дані як веб-застосунку так і мобільному додатку. Також був реалізований інтерфейс побудови і редагування сцен і завдань у віртуальній реальності у веб-застосунку, розроблене спеціальне API для обміну даними із сервером.
2. Створено мобільний додаток для смартфонів, у якому за допомогою окулярів віртуальної реальності та спеціального контролера можна вибирати, проходити завдання, які пов’язані із курсами з побудови розумних будинків. Розроблено інтерфейс взаємодії людини із віртуальною реальністю та можливість додавання нових пристроїв до додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Claudia M. Augmented Reality and Virtual Reality. Empowering Human, Place and Business / M. Claudia, T. Junk – Springer, 2017 – 78p.
2. Yoder P. Mounting Optics in Optical Instruments / P. Yoder – Press Monograph, 2008 – 782p.
3. Linowes J. Cardboard VR Projects for Android / J. Linowes, M. Schoen – Packt, 2016 – 386p.
4. Hocking J. Unity in Action: Multiplatform game development in C# / J. Hocking – Manning, 2018 – 400p.
5. Smith M. Unity 2018 Cookbook: Over 160 recipes to take your 2D and 3D game development to the next level, 3rd Edition / M. Smith – Packt, 2018 - 794p.
6. Doran J. Unity 2018 Shaders and Effects Cookbook: Transform your game into a visually stunning masterpiece with over 70 recipes / J. Doran – Packt, 2018 – 392p.
7. Finnegan T. Learning Unity Android Game Development / T. Finnegan – Packt, 2015 – 346p.
8. Linowes J. Unity Virtual Reality Projects: Explore the world of virtual reality by building immersive and fun VR projects using Unity 3D / J. Linowes – Packt, 2015 – 286p.
9. Richter J. CLR via C# / J. Richter – Microsoft Press, 2012 – 896p.
10. Booch G. Object Oriented Design: With Applications / G.Booch, J.Grady – Benjamin Cummings, 1991 – 209p
11. Booch G. Object-Oriented Analysis and Design with applications / G.Booch, J.Grady – Benjamin Cummings, 1998 – 209p.
12. Albarahi B. C# 7.0 in a Nutshell: The Definitive Reference / B. Albarahi – O`Reilly, 2017 – 1088 p.
13. Troelsen A. Pro C# 7: With .NET and .NET Core / A. Troelsen, P. Japikse – Apress, 2017 – 1372p.

ДОДАТОК А

Система інтеграції елементів віртуальної реальності в освітню WEB-систему

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б

Аркушів 2

Київ 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б	Записка.docx	Текстова частина дипломної роботи
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б 12-1	houseVR.apk	Основний компонент

ДОДАТОК Б

Система інтеграції елементів віртуальної реальності в освітню WEB-систему

Текст програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б 12-1

Аркушів 27

Київ 2019

Файл AnimationInfo.cs:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class AnimationInfo : JsonReady
{
    public string stepID;
    public string lessonID;
    public string courseID;
    public string courseName;
    public string lessonName;

    public void FromJson(string json)
    {
        AnimationInfo animationInfo = JsonUtility.FromJson<AnimationInfo>(json);
        stepID = animationInfo.stepID;
        lessonID = animationInfo.lessonID;
        courseID = animationInfo.courseID;
        courseName = animationInfo.courseName;
        lessonName = animationInfo.lessonName;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

[System.Serializable]
public class AvailableAnimationsInfo : JsonReady
{
    public List<AnimationInfo> availableAnimations = new List<AnimationInfo>();


    public void FromJson(string json)
    {
        AvailableAnimationsInfo availableAnimationsInfo = JsonUtility.FromJson<AvailableAnimationsInfo>(json);
        availableAnimations = availableAnimationsInfo.availableAnimations;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

[System.Serializable]
public class AvailableLessonsInfo : JsonReady
{
    public List<AnimationInfo> availableLessons = new List<AnimationInfo>();


    public void FromJson(string json)
    {
        AvailableLessonsInfo availableLessonsInfo = JsonUtility.FromJson<AvailableLessonsInfo>(json);
        availableLessons = availableLessonsInfo.availableLessons;
    }
```

```
    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Events;
using UnityEngine.Networking;

public delegate void RequestCallback(string text);

public delegate void CallbackVoid();

public delegate void CallbackInt(int n);

public delegate void CallbackString(string str);

public delegate void PlayerUpdate(Player player);
public class Api
{
    public const bool IS_DEBUG = false;

    public SignalVoid LogInSuccess = new SignalVoid();
    public SignalVoid RequestError = new SignalVoid();

    public PlayerUpdate OnPlayerUpdate;

    private const string API_URL = "https://diplomafe.herokuapp.com/";
    private const string LOGIN_TOKEN_PATH = "user/";
    private const string LOGIN_PATH = "user/token";
    private const string ANIMATIONS_PATH = "";
    private const string LESSONS_PATH = "";
    private const string ANIMATION_PATH = "";
    private const string LESSON_PATH = "";
    private Player _currentPlayer;

    public Player CurrentPlayer
    {
        get { return _currentPlayer; }
        private set
        {
            _currentPlayer = value;
            OnPlayerUpdate?.Invoke(_currentPlayer);
        }
    }

    public bool StartApi()
    {
        if (PlayerPrefs.HasKey("Token"))
        {
            StartSendingRequest(LOGIN_TOKEN_PATH + PlayerPrefs.GetString("Token"), OnLogIn);
            return true;
        }

        return false;
    }

    public void LogIn(string email, string password)
    {
        LogInInfo logInInfo = new LogInInfo {email = email, password = password};
```

```
    StartSendingRequest(LOGIN_PATH, logInInfo, OnLogIn);
}

private void OnLogIn(string response)
{
    PlayerShortInfo playerShortInfo = new PlayerShortInfo();
    playerShortInfo.FromJson(response);

    CurrentPlayer = new Player(playerShortInfo.name);
    PlayerPrefs.SetString("Token", playerShortInfo.token);

    LogInSuccess.Invoke();

    FetchAnimations();
    FetchLessons();
}

public void FetchAnimation(string id)
{
    if (IS_DEBUG)
    {
        string animation = Resources.Load<TextAsset>("Animation").ToString();
        OnFetchAnimation(animation);
    }
    else
    {
        StartSendingRequest(ANIMATION_PATH, OnFetchAnimation);
    }
}

private void OnFetchAnimation(string json)
{
    OneLessonInfo oneLessonInfo = new OneLessonInfo();
    oneLessonInfo.FromJson(json);
    Player player = CurrentPlayer;
    player.oneLessonInfo = oneLessonInfo;
    CurrentPlayer = player;
}

public void FetchLesson(string id)
{
    if (IS_DEBUG)
    {
        string lesson = Resources.Load<TextAsset>("Lesson").ToString();
        OnFetchLesson(lesson);
    }
    else
    {
        StartSendingRequest(LESSON_PATH, OnFetchLesson);
    }
}

private void OnFetchLesson(string json)
{
    OneLessonInfo oneLessonInfo = new OneLessonInfo();
    oneLessonInfo.FromJson(json);
    Player player = CurrentPlayer;
    player.oneLessonInfo = oneLessonInfo;
    CurrentPlayer = player;
}

private void FetchAnimations()
{
```

```
    if (IS_DEBUG)
    {
        string animations = Resources.Load<TextAsset>("Animations").ToString();
        OnFetchAnimations(animations);
    }
    else
    {
        StartSendingRequest(ANIMATIONS_PATH, OnFetchAnimations);
    }
}

private void FetchLessons()
{
    if (IS_DEBUG)
    {
        string lessons = Resources.Load<TextAsset>("Lessons").ToString();
        OnFetchLessons(lessons);
    }
    else
    {
        StartSendingRequest(LESSONS_PATH, OnFetchLessons);
    }
}

private void OnFetchLessons(string json)
{
    AvailableLessonsInfo availableLessonsInfo = new AvailableLessonsInfo();
    availableLessonsInfo.FromJson(json);

    Player player = CurrentPlayer;
    player.availableLessonsInfo = availableLessonsInfo;
    CurrentPlayer = player;
}

private void OnFetchAnimations(string json)
{
    AvailableAnimationsInfo availableAnimationsInfo = new AvailableAnimationsInfo();
    availableAnimationsInfo.FromJson(json);

    Player player = CurrentPlayer;
    player.availableAnimationsInfo = availableAnimationsInfo;
    CurrentPlayer = player;
}

private void StartSendingRequest(string path, RequestCallback callback)
{
    EachSecond.Instance.StartCoroutine(SendGetRequest(path, callback));
}

private void StartSendingRequest(string path, JsonReady json, RequestCallback callback)
{
    string jsonStr = json.ToJson();
    EachSecond.Instance.StartCoroutine(SendPostRequest(path,  jsonStr, callback));
}

IEnumerator SendGetRequest(string path, RequestCallback callback)
{
    UnityWebRequest www = UnityWebRequest.Get(API_URL + path);
    yield return www.SendWebRequest();

    if (www.isNetworkError || www.isHttpError)
    {
        Debug.Log(www.error);
```

```
            OnRequestError();
        }
        else
        {
            yield return null;
            callback?.Invoke(www.downloadHandler.text);
        }
    }

    IEnumerator SendPostRequest(string path, string json, RequestCallback callback)
    {
        var jsonBinary = System.Text.Encoding.UTF8.GetBytes(json);

        DownloadHandlerBuffer downloadHandlerBuffer = new DownloadHandlerBuffer();

        UploadHandlerRaw uploadHandlerRaw = new UploadHandlerRaw(jsonBinary);
        uploadHandlerRaw.contentType = "application/json";

        UnityWebRequest www =
            new UnityWebRequest(API_URL + path, "POST", downloadHandlerBuffer, uploadHandlerRaw);
        yield return www.SendWebRequest();

        if (www.isNetworkError || www.isHttpError)
        {
            Debug.Log(www.error);
            OnRequestError();
        }
        else
        {
            callback?.Invoke(www.downloadHandler.text);
        }
    }

    private void OnRequestError()
    {
        RequestError.Invoke();
    }

    private static Api _instance;
    public static Api Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new Api();
            }

            return _instance;
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public interface JsonReady
{
    void FromJson(string json);
    string ToJson();
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
public class LogInInfo : JsonReady
{
    public string email;
    public string password;


    public void FromJson(string json)
    {
        LogInInfo logInInfo = JsonUtility.FromJson<LogInInfo>(json);
        email = logInInfo.email;
        password = logInInfo.password;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

[System.Serializable]
public class PlayerShortInfo : JsonReady
{
    public string email;
    public string token;
    public string name;


    public void FromJson(string json)
    {
        PlayerShortInfo playerShortInfo = JsonUtility.FromJson<PlayerShortInfo>(json);
        email = playerShortInfo.email;
        token = playerShortInfo.token;
        name = playerShortInfo.name;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Serializable]
public class OneLessonInfo : JsonReady
{
    public string stepID;
    public string lessonID;
    public string courseID;
    public int unitType;
    public List<SensorInfo> sensors = new List<SensorInfo>();
    public List<TableStepInfo> tableStep = new List<TableStepInfo>();
    public List<RoomStepInfo> roomStep = new List<RoomStepInfo>();
    public List<int> actions;
```

```
    public void FromJson(string json)
    {
        OneLessonInfo oneLessonInfo = JsonUtility.FromJson<OneLessonInfo>(json);
        stepID = oneLessonInfo.stepID;
        lessonID = oneLessonInfo.lessonID;
        courseID = oneLessonInfo.courseID;
        sensors = oneLessonInfo.sensors;
        tableStep = oneLessonInfo.tableStep;
        roomStep = oneLessonInfo.roomStep;
        unitType = oneLessonInfo.unitType;
        actions = oneLessonInfo.actions;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

[Serializable]
public class SensorInfo : JsonReady
{
    public int sensor;
    public int amount;

    public void FromJson(string json)
    {
        SensorInfo sensorInfo = JsonUtility.FromJson<SensorInfo>(json);
        sensor = sensorInfo.sensor;
        amount = sensorInfo.amount;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

[Serializable]
public class TableStepInfo : JsonReady
{
    public int sensor;
    public int sensorPin;
    public int otherSensor;
    public int otherSensorPin;
    public bool pinToMainUnit;

    public void FromJson(string json)
    {
        TableStepInfo tableStepInfo = JsonUtility.FromJson<TableStepInfo>(json);
        sensor = tableStepInfo.sensor;
        sensorPin = tableStepInfo.sensorPin;
        otherSensor = tableStepInfo.otherSensor;
        otherSensorPin = tableStepInfo.otherSensorPin;
        pinToMainUnit = tableStepInfo.pinToMainUnit;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}
```

```
[Serializable]
public class RoomStepInfo : JsonReady
{
    public int sensor;
    public int placeId;

    public void FromJson(string json)
    {
        RoomStepInfo roomStepInfo = JsonUtility.FromJson<RoomStepInfo>(json);
        sensor = roomStepInfo.sensor;
        placeId = roomStepInfo.placeId;
    }

    public string ToJson()
    {
        return JsonUtility.ToJson(this);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ActionSignalsHolder
{
    public SignalVoid OnToggleDayNight = new SignalVoid();
    public SignalVoid OnToggleLightController = new SignalVoid();
    public SignalVoid OnToggleTemperature = new SignalVoid();
    public SignalVoid OnToggleHumidity = new SignalVoid();
    public SignalVoid OnToggleGas = new SignalVoid();
    public SignalVoid OnWalking = new SignalVoid();
    public SignalVoid OnSayPhrase = new SignalVoid();
    public SignalVoid OnToggleCounter = new SignalVoid();
    public SignalVoid OnSetClock = new SignalVoid();
    public SignalVoid OnEnterHome = new SignalVoid();
    public SignalVoid OnEnterHomeWithCard = new SignalVoid();
    public SignalVoid OnEnableSignal = new SignalVoid();
    public SignalVoid OnDisableSignal = new SignalVoid();
    public SignalVoid OnSpeedTime = new SignalVoid();

    private static ActionSignalsHolder _instance;
    public static ActionSignalsHolder Instance => _instance ?? (_instance = new ActionSignalsHolder());
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LightsController : ReusableObject
{
    public List<GameObject> lights;

    protected override void On()
    {
        SetActiveLight(true);
    }

    protected override void Off()
    {
        SetActiveLight(false);
    }

    void SetActiveLight(bool active)
```

```
    {
        foreach (var lightObject in lights)
        {
            lightObject.SetActive(active);
        }
    }
}

using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[Serializable]
public class PlaceAndObjectLink
{
    public SensorPlace sensorPlace;
    public ReusableObject reusableObject;
}

public class SensorsManager : MonoBehaviour
{
    private readonly Dictionary<int, Sensor> sensors = new Dictionary<int, Sensor>();
    public Transform spawnPoint;

    public List<PlaceAndObjectLink> placeAndObjectLinks;

    private readonly Dictionary<int, SensorPlace> sensorPlaces = new Dictionary<int, SensorPlace>();
    private List<ReusableObject> reusableObjects = new List<ReusableObject>();
    private readonly Dictionary<SensorPlace, Sensor> sensorsInPlaces = new Dictionary<SensorPlace, Sensor>();

    public SignalVoid onUse = new SignalVoid();
    public SignalVoid onError = new SignalVoid();
    public SignalVoid onEndedLook = new SignalVoid();

    private Sensor currentSensor;

    private static SensorsManager _instance;

    private void Awake()
    {
        if (_instance == null)
        {
            _instance = this;
        }

        SensorPlace[] sensorPlacesArray = FindObjectsOfType<SensorPlace>();
        foreach (var sensorPlace in sensorPlacesArray)
        {
            sensorPlaces[sensorPlace.sensorId] = sensorPlace;
            sensorPlace.OnSensorClicked.AddListener(() => OnSensorPlaceClicked(sensorPlace));
        }

        ReusableObject[] reusableObjectsArray = FindObjectsOfType<ReusableObject>();
        foreach (var reusableObject in reusableObjectsArray)
        {
            reusableObjects.Add(reusableObject);
            reusableObject.OnReusableObjectClicked.AddListener((on) => {OnReusableObjectClicked(reusableObject, on);});
        }

        SensorPlace.OnSensorPlaceView.AddListener(OnSensorPlaceView);
        SensorPlace.OnSensorPlaceEndedView.AddListener(() => onEndedLook.Invoke());
```

```
    ReusableObject.OnReusableObjectView.AddListener(() => onUse.Invoke());
    ReusableObject.OnReusableObjectEndedView.AddListener(() => onEndedLook.Invoke());
}

void Start()
{
    sensors.Clear();

    List<SensorInfo> sensorInfos = LessonManager.Instance.GetSensorsInfo();
    foreach (var sensorInfo in sensorInfos)
    {
        GameObject sensorGameObject = GlobalStorage.Instance.GetSensorPrefab(sensorInfo.sensor);
        if (sensorGameObject)
        {
            Sensor sensor = sensorGameObject.GetComponent<Sensor>();
            sensors[sensorInfo.sensor] = sensor;
        }
    }
}

public Sensor GetSensor(int id)
{
    if (sensors.ContainsKey(id))
    {
        return sensors[id];
    }

    return null;
}

void OnSensorPlaceView()
{
    if (currentSensor)
    {
        onUse.Invoke();
    }
}

void OnSensorPlaceClicked(SensorPlace sensorPlace)
{
    if (sensorsInPlaces.ContainsKey(sensorPlace) && currentSensor)
    {
        onError.Invoke();
    }
    else if (currentSensor && !sensorsInPlaces.ContainsKey(sensorPlace))
    {
        PlaceSensorOnPlace(sensorPlace);
    }
    else if(!currentSensor && sensorsInPlaces.ContainsKey(sensorPlace))
    {
        currentSensor = sensorsInPlaces[sensorPlace];
        sensorPlace.EnableHelp();
        sensorsInPlaces.Remove(sensorPlace);
        TakeSensorInHand();
    }
}

void PlaceSensorOnPlace(SensorPlace sensorPlace)
{
    sensorsInPlaces[sensorPlace] = currentSensor;
    sensorPlace.DisableHelp();
    Transform sensorTransform = currentSensor.transform;
    Transform sensorPlaceTransform = sensorPlace.transform;
```

```
        PlaceSensorToTransform(sensorTransform, sensorPlaceTransform);

        currentSensor = null;
    }

    void TakeSensorInHand()
    {
        if (!currentSensor)
        {
            return;
        }

        Transform sensorTransform = currentSensor.transform;
        PlaceSensorToTransform(sensorTransform, spawnPoint);
    }

    void PlaceSensorToTransform(Transform sensorTransform, Transform other)
    {
        sensorTransform.parent = other;
        sensorTransform.localPosition = Vector3.zero;
        sensorTransform.localRotation = Quaternion.identity;
    }

    void OnReusableObjectClicked(ReusableObject reusableObject, bool on)
    {
        List<Sensor> sensors = GetSensorsToDetect(reusableObject);
        DetectSensors(sensors, on);
        reusableObject.UseObject();
    }

    void DetectSensors(List<Sensor> sensors, bool on)
    {
        foreach (var sensor in sensors)
        {
            sensor.Detect(on);
        }
    }

    List<Sensor> GetSensorsToDetect(ReusableObject reusableObject)
    {
        List<Sensor> result = new List<Sensor>();
        foreach (var placeAndObjectLink in placeAndObjectLinks)
        {
            if (placeAndObjectLink.reusableObject == reusableObject &&
sensorsInPlaces.ContainsKey(placeAndObjectLink.sensorPlace))
            {
                result.Add(sensorsInPlaces[placeAndObjectLink.sensorPlace]);
            }
        }

        return result;
    }

    public static SensorsManager Instance => _instance;
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ClickableObject : MonoBehaviour
{
```

```
    public SignalVoid onViewOnObject = new SignalVoid();
    public SignalVoid onViewEndedOnObject = new SignalVoid();
    public SignalVoid onClickedOnObject = new SignalVoid();


    private void Awake()
    {
        onViewOnObject.AddListener(OnViewOnObject);
        onViewEndedOnObject.AddListener(OnViewEndedOnObject);
        onClickedOnObject.AddListener(OnClickedOnObject);
    }

    protected virtual void OnViewOnObject()
    {

    }

    protected virtual void OnViewEndedOnObject()
    {

    }

    protected virtual void OnClickedOnObject()
    {

    }
}

using UnityEngine.EventSystems;
using UnityEngine.UI;

public class ClickableUI : ClickableObject
{
    protected override void OnViewOnObject()
    {
        EventSystem.current.SetSelectedGameObject(gameObject);
    }

    protected override void OnViewEndedOnObject()
    {
        EventSystem.current.SetSelectedGameObject(null);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Door : ReusableObject
{
    public Animator animator;

    protected override void On()
    {
        animator.SetBool("On", true);
    }

    protected override void Off()
    {
        animator.SetBool("On", false);
    }
}
```

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ReusableObject : ClickableObject
{
    public static readonly SignalVoid OnReusableObjectView = new SignalVoid();
    public static readonly SignalVoid OnReusableObjectEndedView = new SignalVoid();
    public readonly SignalBool OnReusableObjectClicked = new SignalBool();

    private bool on = false;

    protected override void OnViewOnObject()
    {
        Debug.Log("Look");
        OnReusableObjectView.Invoke();
    }

    protected override void OnClickedOnObject()
    {
        on = !on;
        OnReusableObjectClicked.Invoke(on);
    }

    public void UseObject()
    {
        if (on)
        {
            On();
        }
        else
        {
            Off();
        }
    }

    protected virtual void On()
    {

    }

    protected virtual void Off()
    {

    }

    protected override void OnViewEndedOnObject()
    {
        OnReusableObjectEndedView.Invoke();
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SensorPlace : ClickableObject
{
    public int sensorId;

    public static readonly SignalVoid OnSensorPlaceView = new SignalVoid();
    public static readonly SignalVoid OnSensorPlaceEndedView = new SignalVoid();
    public readonly SignalVoid OnSensorClicked = new SignalVoid();
```

```
    public GameObject helpBox;

    protected override void OnViewOnObject()
    {
        OnSensorPlaceView.Invoke();
    }

    protected override void OnClickedOnObject()
    {
        OnSensorClicked.Invoke();
    }

    public void DisableHelp()
    {
        helpBox.SetActive(false);
    }

    public void EnableHelp()
    {
        helpBox.SetActive(true);
    }

    protected override void OnViewEndedOnObject()
    {
        OnSensorPlaceEndedView.Invoke();
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class AnimationItem : MonoBehaviour
{
    public Text courseName;
    public Text stepName;
    public Button button;

    public void Init(string course, string step)
    {
        courseName.text = course;
        stepName.text = step;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AnimationList : MonoBehaviour
{
    private const int MAX_ITEMS_COUNT = 3;

    private readonly List<AnimationItem> _animationItems = new List<AnimationItem>();
    private RectTransform rectTransform;

    private void Awake()
    {
        rectTransform = GetComponent<RectTransform>();
    }

    public void ShowItems(List<AnimationInfo> infos, GameObject prefab, CallbackString callbackString)
```

```
    {
        foreach (var animationItem in _animationItems)
        {
            GameObject o;
            (o = animationItem.gameObject).SetActive(false);
            Destroy(o);
        }

        _animationItems.Clear();

        for (int i = 0; i < MAX_ITEMS_COUNT; i++)
        {
            SpawnAnimationItem(infos[i], prefab, i, callbackString);
        }
    }

    void SpawnAnimationItem(AnimationInfo info, GameObject prefab, int index, CallbackString callbackString)
    {
        GameObject animationItem = Instantiate(prefab, rectTransform);
        RectTransform animationItemTransform = animationItem.GetComponent<RectTransform>();
        if (animationItemTransform)
        {
            animationItemTransform.anchoredPosition = new Vector2(0, 100 - index * 200);
            AnimationItem item = animationItem.GetComponent<AnimationItem>();

            if (item)
            {
                item.Init(info.courseName, info.lessonName);
                item.button.onClick.AddListener(() => { callbackString(info.stepID); });
            }
        }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GUI : MonoBehaviour
{
    public class Params
    {
        public bool showPointer;
        public bool showJoySticks;
    }

    protected Params _params;

    public virtual void Init(Params args)
    {
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PinItem : MonoBehaviour
{

    public SignalVoid OnSubmitClick = new SignalVoid();
    public Image image;
```

```
    public Text pinNameText;
    public Color unChoosedColor;
    public Color choosedColor;
    public GameObject usedGameObject;

    public void Init(string pinName)
    {
        pinNameText.text = pinName;
    }

    public void Choose()
    {
        image.color = choosedColor;
    }

    public void UnChoose()
    {
        image.color = unChoosedColor;
    }

    public void SetUsed()
    {
        usedGameObject.SetActive(true);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class PinsList : MonoBehaviour
{
   private const int MAX_ITEMS = 10;

    public List<PinItem> pins = new List<PinItem>();
    public RectTransform content;
    public Text sensorNameText;

    private bool active = true;
    private int currentValue;

    private bool locked;

    private void Start()
    {
        InputManager.Instance.OnMovePlayer.AddListener(OnMovePlayer);
        InputManager.Instance.OnMovePlayer.AddListener(OnMovePlayer);
    }

    void OnMovePlayer(Vector2 move)
    {
        if (!active || locked)
        {
            return;
        }

        if (move.x > 0)
        {
            OnUpClick();
        }

        if (move.x < 0)
```

```
    {
        OnDownClick();
    }
}

private IEnumerator Lock()
{
    locked = true;
    yield return new WaitForSeconds(0.1f);
    locked = false;
}

private void Clean()
{
    currentValue = 0;
    foreach (var sensor in pins)
    {
        Destroy(sensor.gameObject);
    }

    pins.Clear();
}

public void ShowItems(List<Pin> pinsInfo, GameObject pinItemPrefab, string sensorName,CallbackInt callback)
{
    Clean();
    sensorNameText.gameObject.SetActive(true);
    sensorNameText.text = sensorName;

    foreach (var pin in pinsInfo)
    {
        GameObject sensorObject = Instantiate(pinItemPrefab, content);
        PinItem pinItem = sensorObject.GetComponent<PinItem>();
        if (pinItem)
        {
            string pinName = pin.name;
            pinItem.Init(pinName);
            pinItem.OnSubmitClick.AddListener(() => { callback(pin.id); });
            pins.Add(pinItem);
        }
    }

    ScrollToCurrentValue();
}

private void OnUpClick()
{
    if (currentValue > 0)
    {
        currentValue--;
    }
    else
    {
        currentValue = pins.Count - 1;
    }

    ScrollToCurrentValue();
}

private void OnDownClick()
{
    if (currentValue < pins.Count - 1)
    {
```

```
            currentValue++;
        }
        else
        {
            currentValue = 0;
        }

        ScrollToCurrentValue();
    }

    private void ScrollToCurrentValue()
    {
        float upEdge = 1f;
        if (currentValue >= MAX_ITEMS)
        {
            upEdge = 1f + (1f / MAX_ITEMS) * (currentValue - MAX_ITEMS + 1);
        }

        for (int i = 0; i < pins.Count; i++)
        {
            pins[i].GetComponent<RectTransform>().anchorMin = new Vector2(0, upEdge - (1f / MAX_ITEMS) * (i + 1));
            pins[i].GetComponent<RectTransform>().anchorMax = new Vector2(1, upEdge - (1f / MAX_ITEMS) * (i));
        }

        foreach (var sensor in pins)
        {
            sensor.UnChoose();
        }

        StartCoroutine(Lock());
        pins[currentValue].Choose();
    }

    public void Activate()
    {
        active = true;
    }

    public void Deactivate()
    {
        active = false;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class SensorItem : MonoBehaviour
{
    public Image image;
    public Text nameText;
    public GameObject choosedObject;
    public GameObject pluggedObject;
    public readonly SignalVoid OnSubmitClick = new SignalVoid();

    public void Init(Sprite sprite, string namestr)
    {
        image.sprite = sprite;
        nameText.text = namestr;
        InputManager.Instance.OnSubmitClick.AddListener(OnSubmitClicked);
    }
```

```
    private void OnSubmitClicked()
    {
        OnSubmitClick.Invoke();
    }

    public void Choose()
    {
        choosedObject.SetActive(true);
    }

    public void UnChoose()
    {
        choosedObject.SetActive(false);
    }

    public void SetPlugged()
    {
        pluggedObject.SetActive(true);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.XR;

public class SensorsList : MonoBehaviour
{
    private const int MAX_ITEMS = 3;

    public List<SensorItem> sensors = new List<SensorItem>();
    public RectTransform content;

    private bool active = true;
    private int currentValue;

    private bool locked;

    private void Start()
    {
        InputManager.Instance.OnMovePlayer.AddListener(OnMovePlayer);
        InputManager.Instance.OnMovePlayer.AddListener(OnMovePlayer);
    }

    void OnMovePlayer(Vector2 move)
    {
        if (!active || locked)
        {
            return;
        }

        if (move.x > 0)
        {
            OnUpClick();
        }

        if (move.x < 0)
        {
            OnDownClick();
        }
    }
```

```
private IEnumerator Lock()
{
    locked = true;
    yield return new WaitForSeconds(0.1f);
    locked = false;
}

private void Clean()
{
    currentValue = 0;
    foreach (var sensor in sensors)
    {
        Destroy(sensor.gameObject);
    }

    sensors.Clear();
}

public void ShowItems(List<SensorInfo> sensorInfos, int mainUnit, GameObject sensorItemPrefab, CallbackInt callback)
{
    Clean();

    int index = 0;
    foreach (var sensor in sensorInfos)
    {
        for (int i = 0; i < sensor.amount; i++)
        {

            GameObject sensorObject = Instantiate(sensorItemPrefab, content);
            SensorItem sensorItem = sensorObject.GetComponent<SensorItem>();
            if (sensorItem)
            {
                Sprite sprite = GlobalStorage.Instance.GetSensorSprite(sensor.sensor);
                string sensorName = GlobalStorage.Instance.GetSensorName(sensor.sensor);
                sensorItem.Init(sprite, sensorName);
                sensorItem.OnSubmitClick.AddListener(() => { callback(sensor.sensor); });
                sensors.Add(sensorItem);
            }
            index++;
        }
    }

    ScrollToCurrentValue();
}

private void OnUpClick()
{
    if (currentValue > 0)
    {
        currentValue--;
    }
    else
    {
        currentValue = sensors.Count - 1;
    }

    ScrollToCurrentValue();
}

private void OnDownClick()
{
    if (currentValue < sensors.Count - 1)
```

```
        {
            currentValue++;
        }
        else
        {
            currentValue = 0;
        }

        ScrollToCurrentValue();
    }

    private void ScrollToCurrentValue()
    {
        float upEdge = 1f;
        if (currentValue >= MAX_ITEMS)
        {
            upEdge = 1f + (1f / MAX_ITEMS) * (currentValue - MAX_ITEMS + 1);
        }

        for (int i = 0; i < sensors.Count; i++)
        {
            sensors[i].GetComponent<RectTransform>().anchorMin = new Vector2(0, upEdge - (1f / MAX_ITEMS) * (i + 1));
            sensors[i].GetComponent<RectTransform>().anchorMax = new Vector2(1, upEdge - (1f / MAX_ITEMS) * (i));
        }

        foreach (var sensor in sensors)
        {
            sensor.UnChoose();
        }

        StartCoroutine(Lock());
        sensors[currentValue].Choose();
    }

    public void Activate()
    {
        active = true;
    }

    public void Deactivate()
    {
        active = false;
    }

    private void OnStartButtonClicked()
    {
        active = !active;
    }

}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ViewGUI : GUI
{
    public GameObject pointer;

    public override void Init(Params args)
    {
        _params = args;
```

```
        pointer.SetActive(_params.showPointer);
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ActionManager
{
    public class ActionParams
    {
        public int id;
        public string text;
        public CallbackVoid onActionEnded;
    }
    private readonly List<Action> registeredActions = new List<Action>();
    private List<int> actions = new List<int>();
    public readonly SignalVoid onActionEnded = new SignalVoid();

    private void RegisterActions()
    {

    }

    private void RegisterAction(Action action)
    {
        action._params.onActionEnded = OnActionEnded;
        registeredActions.Add(action);
    }

    void OnActionEnded()
    {
        Action lastAction = GetNextAction();
        if (lastAction != null)
        {
            actions.RemoveAt(0);
        }

        onActionEnded.Invoke();
    }

    public void Init()
    {
        Api.Instance.OnPlayerUpdate += OnPlayerUpdate;
        RegisterActions();
    }

    private void OnPlayerUpdate(Player player)
    {
        if (player.oneLessonInfo != null)
        {
            actions = new List<int> (player.oneLessonInfo.actions);
        }
    }

    public string GetNextActionText()
    {
        Action action = GetNextAction();
        if (action != null)
        {
            return action._params.text;
        }
```

```
        return null;
    }

    private Action GetNextAction()
    {
        if (actions.Count > 0)
        {
            Action action = GetActionById(actions[0]);
            if (action != null)
            {
                return action;
            }
        }

        return null;
    }

    private Action GetActionById(int id)
    {
        foreach (var action in registeredActions)
        {
            if (action._params.id == id)
            {
                return action;
            }
        }

        return null;
    }

    private static ActionManager _instance;

    public static ActionManager Instance => _instance ?? (_instance = new ActionManager());
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class LessonManager
{
    private List<AnimationInfo> availableAnimationsInfo = new List<AnimationInfo>();
    private List<AnimationInfo> availableLessonsInfo = new List<AnimationInfo>();

    private OneLessonInfo currentLesson;
    private bool isAnimated;

    private List<TableStepInfo> connections = new List<TableStepInfo>();

    public void Init()
    {
        Api.Instance.OnPlayerUpdate += OnPlayerUpdate;
    }

    void OnPlayerUpdate(Player player)
    {
        availableAnimationsInfo = player.availableAnimationsInfo.availableAnimations;
        availableLessonsInfo = player.availableLessonsInfo.availableLessons;
        currentLesson = player.oneLessonInfo;
        connections.Clear();
        if (currentLesson != null)
        {
```

```
            connections = currentLesson.tableStep;
        }
    }

    public void StartAnimation(string animationId)
    {
        Api.Instance.FetchAnimation(animationId);
        isAnimated = true;
    }

    public void StartLesson(string lessonId)
    {
        Api.Instance.FetchLesson(lessonId);
        isAnimated = false;
    }

    public bool ApplyConnectionIfCan(int pinFrom, int pinTo, int sensorOne, int sensorTwo, bool isMainUnit)
    {
        TableStepInfo tableStepInfo = null;
        foreach (var connection in connections)
        {
            if ((connection.sensor == sensorOne && connection.otherSensor == sensorTwo &&
                 connection.sensorPin == pinFrom && connection.otherSensorPin == pinTo ||
                 connection.sensor == sensorTwo && connection.otherSensor == sensorOne &&
                 connection.sensorPin == pinTo && connection.otherSensorPin == pinFrom) &&
                 connection.pinToMainUnit == isMainUnit)
            {
                tableStepInfo = connection;
            }
        }

        if (tableStepInfo != null)
        {
            connections.Remove(tableStepInfo);
            return true;
        }

        return false;
    }

    public bool CanApplyConnections()
    {
        return connections.Count > 0;
    }

    public List<SensorInfo> GetSensorsInfo()
    {
        return currentLesson?.sensors;
    }

    public int GetMainUnit()
    {
        if (currentLesson != null)
        {
            return currentLesson.unitType;
        }

        return -1;
    }

    public void EndCurrentAnimation()
    {
    }
```

```
    public List<AnimationInfo> GetAnimations()
    {
        return availableAnimationsInfo;
    }

    public List<AnimationInfo> GetLessons()
    {
        return availableLessonsInfo;
    }

    public bool Animated()
    {
        return isAnimated;
    }

    private static LessonManager _instance;

    public static LessonManager Instance
    {
        get
        {
            if (_instance == null)
            {
                _instance = new LessonManager();
            }

            return _instance;
        }
    }
}
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Player
{
    private string _name;
    public AvailableAnimationsInfo availableAnimationsInfo = new AvailableAnimationsInfo();
    public AvailableLessonsInfo availableLessonsInfo = new AvailableLessonsInfo();
    public OneLessonInfo oneLessonInfo;

    public Player(string name)
    {
        _name = name;
    }

    public string Name
    {
        get { return _name; }
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerController : MonoBehaviour
{
    private CharacterController _characterController;
    public Transform playerCamera;
    public Transform vrPlayer;
    public float speed;
```

```
    private Transform _transform;
    private Vector2 _destination = new Vector2();
    private bool canMove = true;

    // Start is called before the first frame update
    void Start()
    {
        _transform = transform;
        InputManager.Instance.OnMovePlayer.AddListener(OnMovePlayer);
        _characterController = GetComponent<CharacterController>();
    }

    void OnMovePlayer(Vector2 dest)
    {
        _destination = dest;
    }

    private void Update()
    {
        Move();
        Rotate();
    }

    private void Move()
    {
        if (!canMove)
        {
            return;
        }

        Vector3 horizontalMovement = _transform.forward * speed * Time.deltaTime * _destination.x;
        Vector3 verticalMovement = _transform.right * speed * Time.deltaTime * _destination.y;

        _characterController.SimpleMove(horizontalMovement + verticalMovement);

        vrPlayer.transform.position = _transform.position;
    }

    private void Rotate()
    {
        Vector3 rotation = _transform.rotation.eulerAngles;
        Vector3 playerCameraRotation = playerCamera.rotation.eulerAngles;
        rotation.y = playerCameraRotation.y;
        _transform.rotation = Quaternion.Euler(rotation);
    }

    public void DisableMoving()
    {
        canMove = false;
    }

    public void EnableMoving()
    {
        canMove = true;
    }
}

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
```

```
public class PlayerInfo
{
    public string Username;
    public string Token;
}
```

ДОДАТОК В

Система інтеграції елементів віртуальної реальності в освітню WEB-систему

Опис програми

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б 13-1

Аркушів 10

Київ 2019

АНОТАЦІЯ

Додаток містить опис основного компоненту системи інтеграції елементів віртуальної реальності в освітню WEB-систему, а саме:

Забезпечення обмеження доступу до функціоналу додатку;

Загрузка сцен та завдань у віртуальній реальності із зовнішнього джерела, а саме: із клієнтського веб-застосунку;

Реалізація зручного інтерфейсу для доступу до функціоналу додатку.

Додаток розроблений за допомогою мови програмування C# за допомогою технологій Unity3D та Cardboard, у середовищі програмування IntelliJ Rider.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку міститься опис основного компоненту системи інтеграції елементів віртуальної реальності в освітню WEB-систему, що виконує деякі із завдань, поставлених в розділі 1. У додатку Б міститься програмний код компоненту.

Система інтеграції елементів віртуальної реальності в освітню WEB-систему, працює на мобільних платформах, та для роботи потребує смартфон із версією Android не менше, ніж 7.0.

Система інтеграції елементів віртуальної реальності розроблена за допомогою мови програмування C# та технологій Unity3D та Cardboard, у середовищі програмування IntelliJ Rider.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Додаток надає функціональність для взаємодії із віртуальною реальністю.

— завантаження завдань;

— виконання завдань у віртуальній реальності.

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Згідно з архітектурою, додаток складається з 4 головних модулів:

— крок авторизації;

— крок вибору завдання;

— крок з’єднання пристроїв;

— крок розташування пристроїв.

Крок авторизації потрібен, щоб користувач міг ввійти в систему. На цьому кроці вимкнена віртуальна реальність, щоб користувачу було легше ввести свої дані.

Крок вибору завдання включає в себе завантаження та перегляд усіх доступних завдань та вибір завдання для виконання.

Крок з’єднання пристроїв містить в собі вибір потрібних пристроїв та з’єднання відповідних роз’ємів між собою.

Крок розташування пристроїв містить в собі вибір потрібних пристроїв та розташування їх на спеціальні місця.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Для використання додатку користувач повинен мати обліковий запис у веб-системі. Також йому потрібен смартфон із Android версії не мен

ВИКЛИК І ЗАВАНТАЖЕННЯ

Система інтеграції елементів віртуальної реальності в освітню WEB-систему потребує інсталяції на смартфон за допомогою спеціального apk файлу.

ВХІДНІ ДАНІ

Вхідна інформація для додатку:

а) Пароль та логін користувача.

ВИХІДНІ ДАНІ

Вихідна інформація додатку:

а) Результат виконання завдання.

ДОДАТОК Г

Інтеграція елементів віртуальної реальності в освітню WEB-систему

Тези на конференцію “Сучасні проблеми наукового забезпечення енергетики”

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТІ51184_18Б

Аркушів 3

Київ 2019

Стрімкий розвиток сучасного інформаційного суспільства нерозривно пов’язаний з гнучким оновленням та інтенсивним переосмисленням системи університетської освіти, якість та ефективність якої базується на глибокому зануренні як студентів, так і викладачів у цифрове інформаційне середовище.

Віртуальна реальність — це тривимірний комп'ютерний простір, який може досліджувати користувач. З предметами можна взаємодіяти, а на реальність — впливати зсередини симуляції. Технології для створення віртуальної реальності розроблялися ще з п'ятдесятих років минулого століття. Але саме останнє десятиліття можна назвати періодом розквіту VR-технологій.

Сьогодні для того, щоб почати взаємодіяти з віртуальною реальністю, не потрібно витрачати величезну кількість грошей. Технологія з кожним роком стає все більш доступною. Прилади для VR можна купити за невелику суму (навіть з Google Cardboard можна отримати віртуальний досвід), та доторкнутися до них на виставках.

Використання віртуальної реальності відкриває багато нових можливостей в навчанні та освіті, які є доволі складними, затратними за часом або дорого коштують при традиційних підходах. Виокремлюють п’ять основних переваг застосування VR технологій: в освіті: наочність, безпека, залучення, фокусування, віртуальні заняття [1].

Підвищення ефективності навчання з використанням технологій віртуальної реальності обумовлене також тим, що заняття з використанням сучасних технологій викликають великий інтерес, результатом чого стає підсилення навчальної мотивації та активності учнів [2].

Метою роботи є створення системи інтеграції елементів віртульної реальності в освітню WEB-систему моделювання процесів створення розумного будинку. Це дозволить отримати знання про роботу процесів розумного будинку кожному користувачу, так і підготувати спеціалістів в галузі розумних будинків.

Базуючись на проведених дослідженнях, зроблено висновок, що аналогів даної WEB-системи не існує, а також елементи віртуальної реальності сприяють кращому вивченні галузі, тому дана система з елементами VR є актульною.

Використані джерела:

1. 1. Loscos, C., Widenfeld, H., Roussou, M. Meyer, A., Tecchia, F., Drettakis, G., Gallo, E.,Martinez, A. R., Tsingos, N., Chrysanthou, Y., Robert, L., Bergamasco, M., Dettori, A., & Soubra, S.: The CREATE Project: Mixed Reality for Design, Education, and Cultural Heritage with a Constructivist Approach, ISMAR 03, The Second IEEE and ACM International Symposium on Mixed and Augmented Reality, The National Center of Sciences, Tokyo, Japan, Oct. 7 - Oct. 10 (2003).

2. Winn, W.D.: Learning in artificial environments: Embodiment, embeddedness and dynamic adaptation. Technology, Instruction, Cognition and Learning, 1, 87-114 (2003).